Palestine Polytechnic University

College of Administrative Science and Informatics
Information Technology Department

Developing a New Approach for Detecting and Analysing Segmental Genome Duplication in the Genome Project of Lieshmania Parasite

تطوير طريقة جديدة لتحديد وتحليل مناطق التضاعف في الخريطة الجينية لطفيل الليشمانيا

Prepared by:

Mohammad Ismael Amro
Ammar Waleed Herbawi

Project Supervisor

Dr.Yaqoub Ashhab
Dr.Mahmoud Al-saheb

This project is submitted in partial fulfillment of the requirements for the degree of B.Sc. in Information Technology in Palestine Polytechnic University.

June 2006

Abstract

In the world of genomic research, projects aimed at revealing the complete sequence of a given genome are usually cumbersome and tedious. The major problem in these projects is that the long sequence of the genome should pass two processes; First, randomly cutting it into a huge number of short DNA sequences known as WGSRs (Whole Genome Shotgun Reads). Second, thereafter using computer programs to reconstruct the whole genome sequence. The main drawback during the reconstruction process is the small fragments that come from duplicated segment in the original genome. Segmental duplication is a well known phenomenon which refers to blocks of DNA sequence that exist in more than one copy in the genome. The objective of our project is to develop a Windows-based system that can efficiently analyze and delineate segmental duplications in WGSRs databases. The main advantage of this system is its simplicity for users as well as capability to be used for any WGSRs database. Perl was used for developing this system because it is a very strong language in string processing and analysis. In addition, Perl has many bioinformatics modules that can facilitate programming for biological applications.

Dedication

To my parent and family…

To our instructors Dr. Mahmmoh Al-saheb and Yaqoub Al-ashhab .

To our friends at the University…

To the coming generation who will benefit from this project…

To all martyrs who scarify themselves struggling toward freedom,

and to all our teachers, lecturers, and friends.

Project team:

Mohammad

Ammar

Acknowledgement

First we would thank god for giving our the bless of thinking and studying

We would like to acknowledge our gratitude to all those who have helped our in the writing of this project. First and foremost, we wish to express my sincere thanks to Dr.Yaqoub al-Ashhab and Dr.Mahmoud al-saheb for their insightful comments on an earlier version of the project.

Also we are thankful to our friends for offering invaluable suggestions about the project.

Finally, also we are thank our parent for encouraging, and giving the suitable environment

Table of Contents

Chapter Four: System Design

Chapter Five: Implementation and Coding

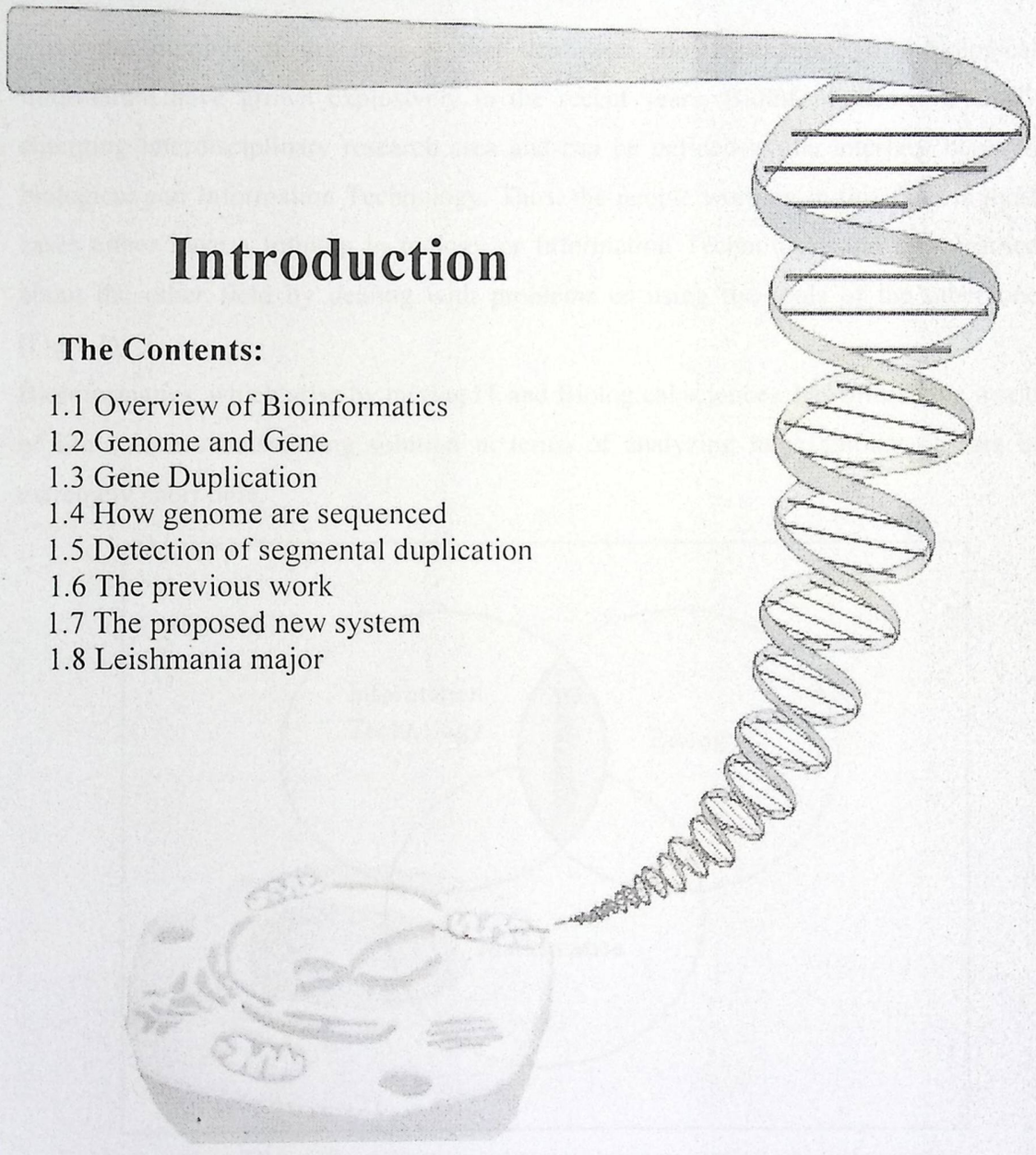
Chapter Six: System Testing

Chapter Seven: System Maintenance

List of Tables

List of Figures

1 Chapter One



Introduction

The Contents:

1.1 Overview for Bioinformatics

Today the world focuses on mixing technologies and deploying them to standardize the process of solving problems. Information Technology is related to all sciences and to all aspects of our life; it's relation to biological sciences is strengthened by Bioinformatics, which is defined as conceptualizing biology in terms of molecules and then applying informatics techniques to understand and organize information associated with these molecules.

As the number of the projects that deal with the organization of biological information have grown explosively in the recent years, Bioinformatics is a newly emerging interdisciplinary research area and can be defined as the interface between biological and Information Technology. Thus, the people working in this field in most cases either have a training in biology or Information Technology, and they learned about the other field by dealing with problems or using the tools of the other one (Fig 1.1).

Bioinformatics, which arise by mixing IT and Biological sciences, has offered the world of life sciences outstanding solution in terms of analyzing huge amount of data in extremely short time.

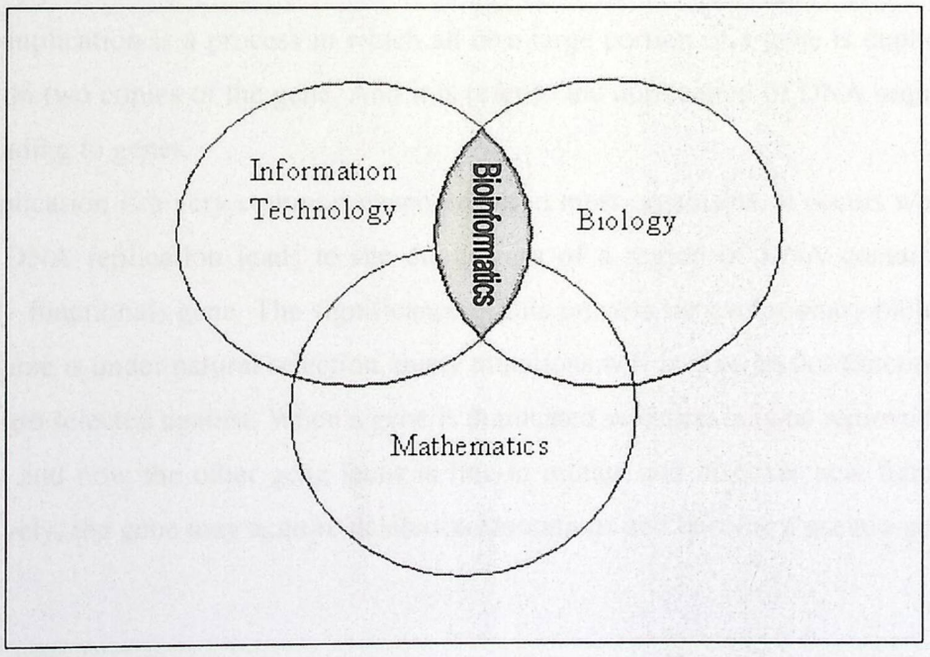


Figure [1.1] The major sciences and technologies constitute bioinformatics

1.2 Genome and Gene

A genome is all of a living things genetic material. It is the entire set of hereditary instructions for building, running, and maintaining an organism, and passing life on to the next generation. The whole shebang.

A G T C C G C G A A T A C A G G C T C G G T

In most living things, the genome is made of a chemical called DNA. The genome contains genes, which are packaged in chromosomes and affect specific characteristics of the organism.

A gene is a small piece of the genome. It is the genetic equivalent of the atom: As an atom is the fundamental unit of matter, a gene is the fundamental unit of heredity. Genes are found on chromosomes and are made of DNA. Different genes determine the different characteristics, or traits, of an organism. In the simplest terms (which are actually too simple in many cases), one gene might determine the color of a bird's feathers, while another gene would determine the shape of its beak.

1.3 Gene Duplication

Gene duplication is a process in which all or a large portion of a gene is duplicated, resulting in two copies of the gene. And it is refer to the duplication of DNA sequences corresponding to genes.

Gene duplication is a very common phenomenon in most organisms. It occurs when an error in DNA replication leads to the duplication of a region of DNA containing a (generally functional) gene. The significance of this process for evolutionary biology is that if a gene is under natural selection, many mutations will lead to loss of functionality and thus are selected against. When a gene is duplicated selection may be removed from one copy and now the other gene locus is free to mutate and discover new functions. Alternatively, the gene may acquire deleterious mutations and become a pseudo-gene. [1]

[1] Brown, T. A. **Genomes**. BIOS Scientific publisher. Oxford, UK 2nd Edition 2002.

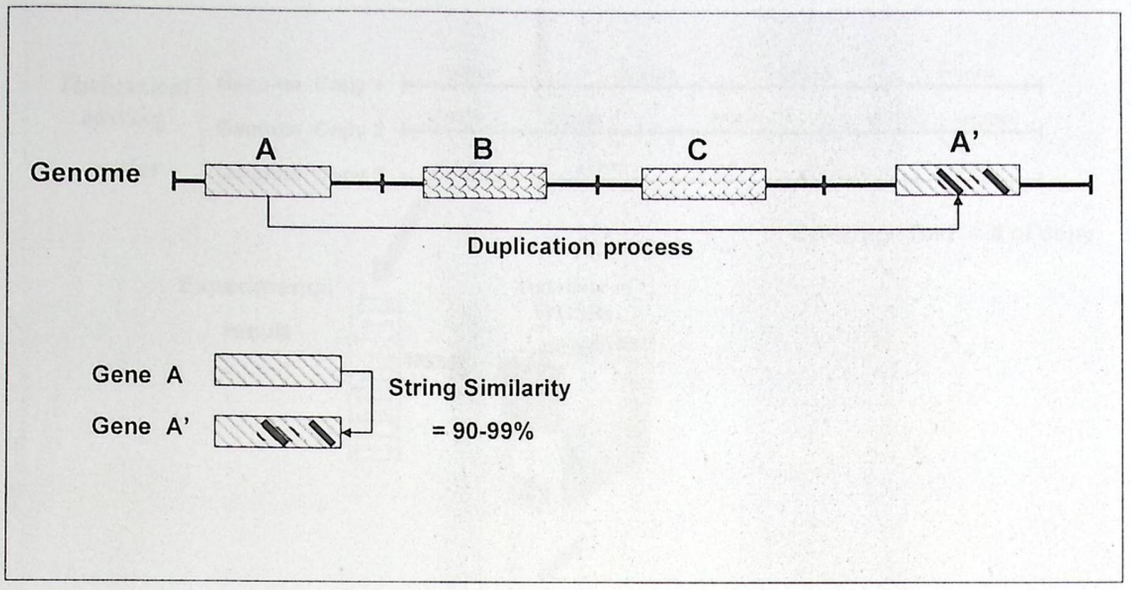


Figure |1.2| Gene Duplication Diagram.

Figure 1.1 shows a hypothetical genome as a single string chromosome that contains several genes (indicated by shaded boxes). The difference in shading between A (old copy) and A' (new copy) represents the new changes that were acquired by the recently duplicated copy A'

1.4 How genome are sequenced

Since the genomes are usually very long single or multiple chromosomes (strings), there is no biological technique to read and determine the whole sequence of a given genome at once. Therefore, biologists usually solve this technical problem by dividing the long chromosome into a large number of small fragments, these fragments are known as WGSRs. The generation of these fragments is totally a random process, which means that the cutting process will generate a random number of WGSRs, each has a random length. After finishing the cutting process, each WGSR is sequenced by a simple biological technique and the order of its nucleotides (characters) is determined. The next step is to find the order of the fragment in relating to each other. This is achieved by haring more than one string of the whole genome there afire, by find the overlapping parts of the different fragment the whole genome is built. This process is call genome ensample by WESR overlapping. [2]

[2] Brown, T. A. **Genomes**. BIOS Scientific publisher. Oxford. UK 2nd Edition 2002.

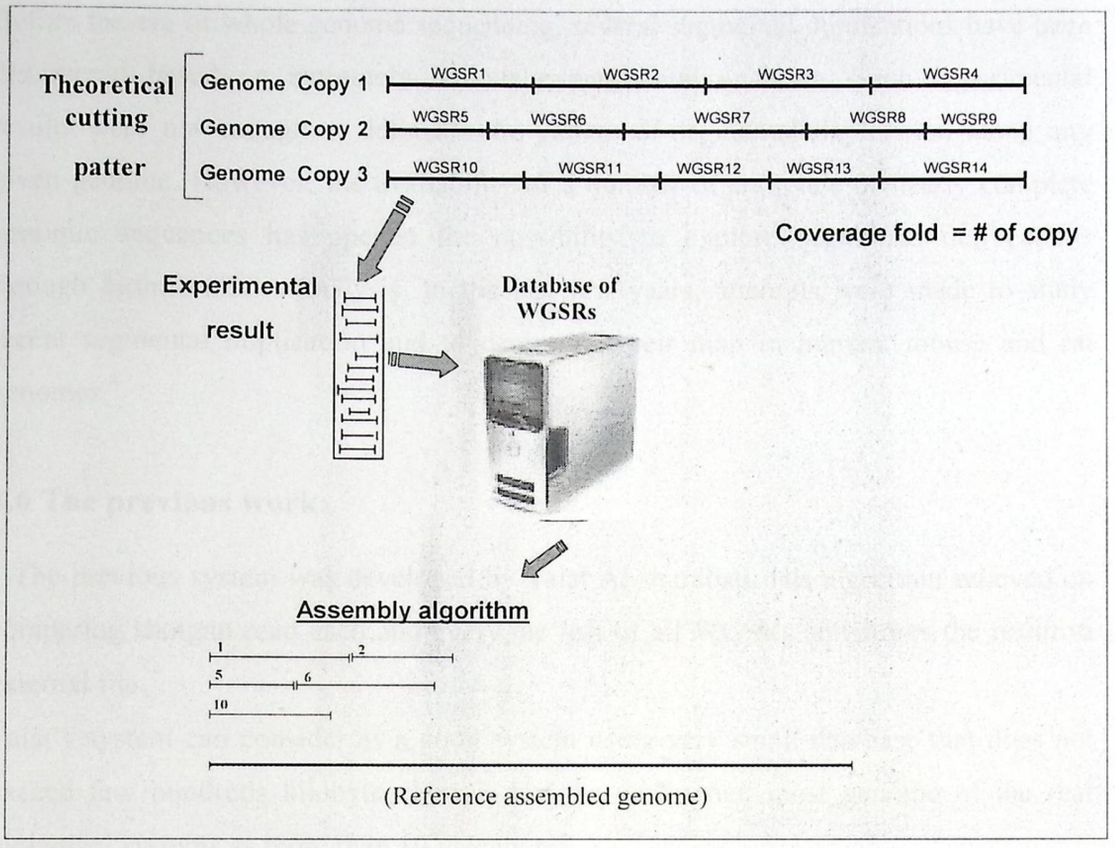


Figure [1.3] Whole Genome Construction and Assembly Process.

It is important to note that the number of copies of the genome (whole string) that is under sequencing and assembly process should be at least two in order to facilitate the process of finding overlapping tips in WGSRs. The number of genome copies is usually referred as coverage fold. For example, when three copies of a given genome are cut down into small fragments (WGSRs) in order to construct and assemble the whole genome, the coverage fold is said to be[3].

1.5 Detection of segmental duplication:

Recent segmental duplications are blocks of genomic DNA sequences that are highly similar (90%-98%) and they range in size from 1kb to 200 kb. segmental duplications can be divided into two classes; inter-chromosomal and intra-chromosomal. Due to their size, these segments can include full and/or partial genes, regulatory sequences and high-copy repeats.[3]

3 Eichler EE. **Recent duplication, domain accretion and the dynamic mutation of the human genome**. Trends Genet. 2001 Nov; 17(11):661-9.

Before the era of whole genome sequencing, several segmental duplications have been discovered based on extremely tedious experimental analyses. Such experimental results were not enough to delineate the pattern of segmental duplication along any given genome. However, the availability of a number of complete or nearly complete genomic sequences has opened the possibility to explore segmental duplications through bioinformatics analysis. In the last few years, attempts were made to study recent segmental duplication and to determine their map in human, mouse and rat genomes.[4]

1.6 The previous work:

The previous system was developed by Talat Al-sharabati. His algorithm relieved on comparing shotgun read each and every the rest of all WGSRs and stores the result in external file.

Talat's system can consider as a good system using very small database that does not exceed few hundreds kilobyte. This is not the case when most genome of the real biological systems as more than 10 megabytes.

In addition, prompted the search for each shotgun read followed by storing and analyzing each result's file would necessity a computer with extraordinary features in term of CPU speed and hard disk capacity.

The interface of the system is not comfortable for the user, because it depends on displaying the output on external files, so this was inconvenient for the user, also the time needed for the output to be displayed was very huge, and sometimes the system became idle when comparing large data.

The above mentioned limitations and drawbacks prompted to propose a totally different approach.

4 Bailey JA, Eichler EE. **Genome-wide detection and analysis of recent segmental duplications within mammalian organisms**. Cold Spring Harb Symp Quant Biol. 2003; 68:115-24.

The following figure shows the previous system works:

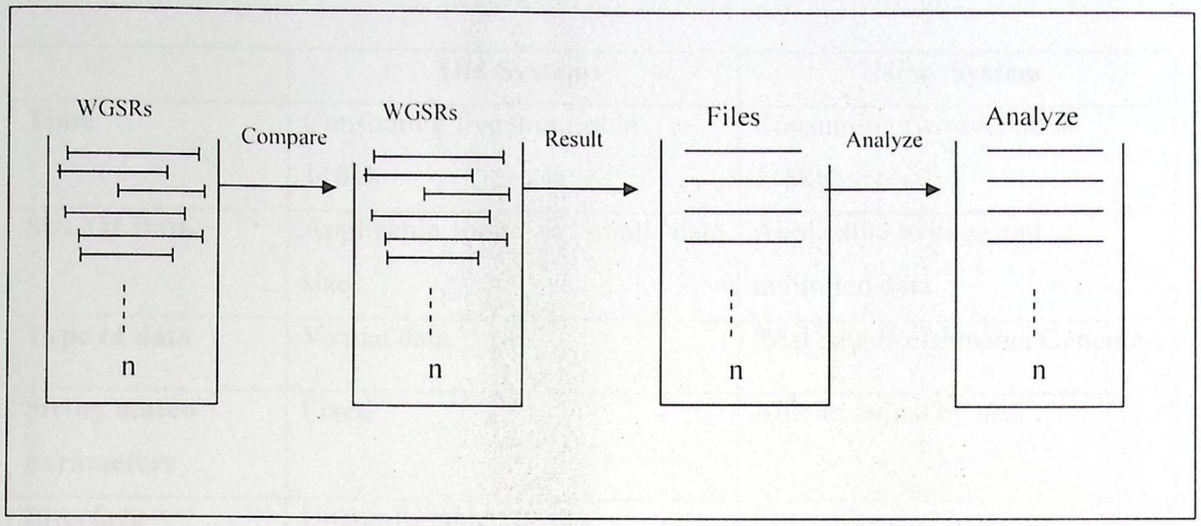


Figure [1.4] Talat system works

1.7 The proposed new system

The new system tries to solve all the problems found in the old system in order to match the user requirement. It will be introduced to test the previous system and validate and improve it. It will also use new algorithm to solve the problem on large real data with short processing time.

This system works by comparing genome assembly reference genome (see fig [1.3]) with all WGSRs, this process make the system works more efficiency and effectively than the old system.

The following figure shows the new system works:

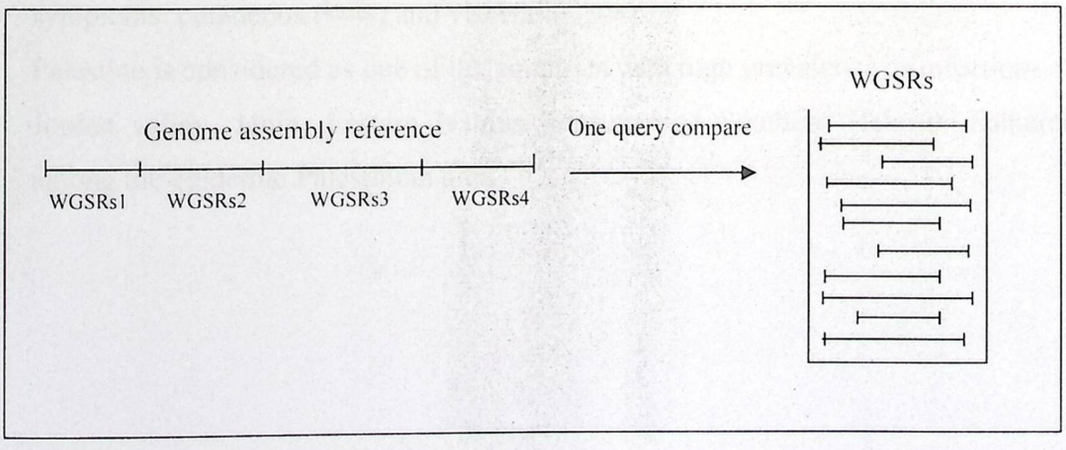


Figure [1.5] The new system works

The following table represents comparison between the new system and the old system:

	Old System	New System
Time	Consuming five minutes in 118kb.	Consuming two second in 118kb.
Size Of Data	Applicable only to small data size.	Applicable to huge and unlimited data.
Type of data	Virtual data	Real data (Leishmania Genome)
String match parameters	Fixed	Able to adjust by user
Interface	Uncomfortable	Convenient

Table [1.1] Comparison Between the Talat's system and the new system

1.8 Leishmania Major (طفيل اللشمانيا)

The Leishmania major is an intracellular pathogen of the immune system targeting macrophages and dendritic cells. The disease Leishmaniasis affects the populations of 34 counties worldwide with symptoms ranging from disfiguring cutaneous and muco-cutaneous lesions that can cause widespread destruction of mucous membranes to visceral disease affecting the haemopoetic organs.

Leishmaniasis currently threatens 220 million men, women and children in 34 countries around the world. Leishmaniasis is parasitic diseases with a wide range of clinical symptoms: cutaneous (جلدية) and visceral (حشوية).

Palestine is considered as one of the countries with high prevalence of infections.

Jordan valley, Jenin, Eastern Nablus, Eastern and Southern Hebron, Tolkarem are among the epidemic Palestinian area.

The following figure shows the areas that infected by Leishmania:

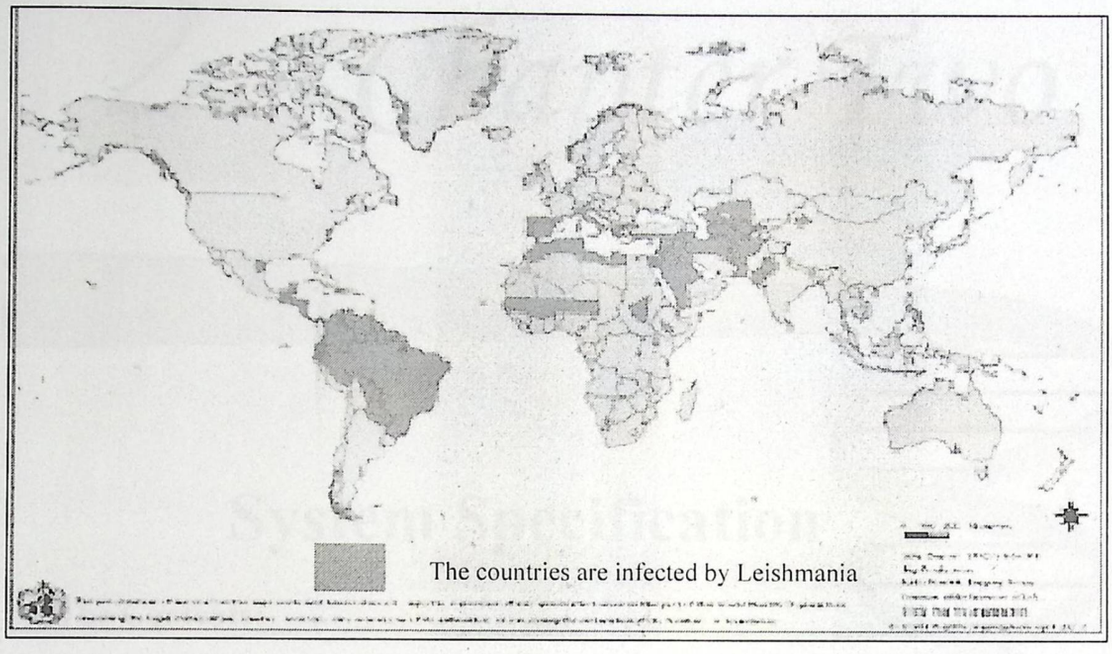
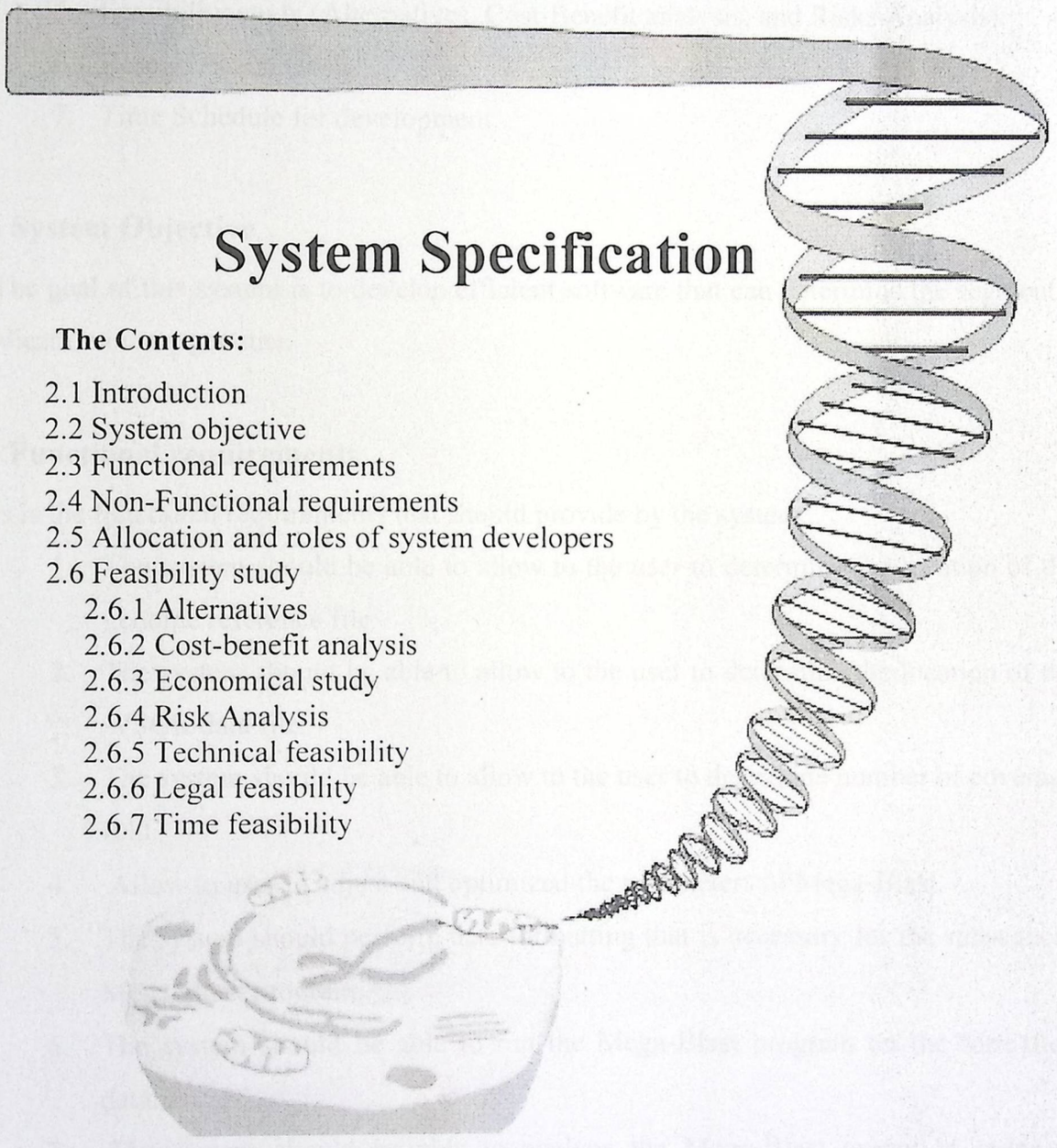


Figure [1.6] The countries are infected by Leishmania

2 Chapter Two



System Specification

The Contents:

2.1 Introduction

In this chapter we will describe the system specifications. The description will include the following topics:

1. System objective.
2. Functional and Non-Functional requirements.
3. Allocation of roles of system developers.
4. Constraints.
5. Feasibility study (Alternatives, Cost-Benefit analysis, and Risks Analysis).
6. Resources and Cost.
7. Time Schedule for development.

2.2 System Objective

The goal of this system is to develop efficient software that can determine the segmental duplication in any genome.

2.3 Functional requirements

This is the functional requirements that should provide by the system:

1. The system should be able to allow to the user to determine the location of the genome reference file
2. The system should be able to allow to the user to determine the location of the WSGR data file.
3. The system should be able to allow to the user to determine number of coverage fold.
4. Allow to user to adjust and optimized the parameters of Mega-Blast.
5. The system should perform data formatting that is necessary for the subsequent Mega-Blast program.
6. The system should be able to run the Mega-Blast program on the formatted data.
7. The system should be able to analyze the Mega-Blast output in order to determine the duplicated segments in the tested genome sequences.

8. Present the result in a biological interpretable style.

2.4 Non-Functional requirements

In this section, all system non-functional requirements are described as well as their specifications. These non-functional requirements include:

- **Product Requirement**
  1. Appearance: attractive and competitive methodology for displaying program interface.
  2. User friendly: attractive and competitive methodology for displaying interface.
  3. Usability: easy and simple for use GUI.
  4. High speed program: The system enables the user to perform all the processes in a short period of time, and take the results of the processes very quickly.
  5. High performance system: The system will provide very high quality output, with less cost.
  6. Portability: running on different operating systems without any problems on the application.
  7. Accuracy: the system must provide a high level of accuracy.

- **Process requirement**

  The system and its explanatory documentation must be delivered on 17th June, 2006.

2.5 Allocation and roles of system developers

1. Leader: responsible of planning, scheduling and controlling flow of system development processes.
2. Programmer: responsible of the system programming, implementation testing, so he must have enough experiences in Perl development environment and graphic design.
3. Software engineer: responsible for the documentation and tracing of the development stages of the software.

4. Interface designer: responsible of the program design and user interface of the system (GUI).

2.6 Feasibility Study

For a system to be developed from scratch, the most important issue is to evaluate its benefits versus its cost. In this section, we will describe the system alternatives that could be adapted; a Cost-Benefit analysis is conducted to justify the decision for developing the system, and an evaluation of the risks that may face the system and the development process.

2.6.1 Alternatives

- **Environment:**

We can use many operating systems to implement this program. But we use the windows system because of its easiness, flexibility and wide distribution.

Perl is a programming language developed by Larry Wall, especially designed for processing text. Because of its strong text processing abilities, Perl has become one of the most popular languages for writing CGI scripts. Perl is an interpretive language, which makes it easy to build and test simple programs. Perl is a popular programming language that is extensively used in areas such as bioinformatics and web programming. Perl has become popular with biologists because it is so well-suited to several bioinformatics tasks and applications.

The following features among the obvious advantage that prompted us to select Perl for our project:

1. Perl programs are not platform dependant
2. Ease of Programming.
3. Free Source.
4. Rapid Prototyping.
5. Easy to learn.
6. Portability, Speed, and Program Maintenance

In addition to its standard environment, Perl has a collection of perl modules that facilitate the development of perl scripts for bioinformatics, which called Bioperl. As such, it does include ready to use programs in the sense that may commercial packages and free web-based interfaces do. On other hand, bioperl does provide reusable perl modules that facilitate writing perl scripts for sequence manipulations, accessing of databases using arrange of data formats and executions and parsing of perl results of various molecular biology programs including Blast, Clustalw, TCoffee, Genescan, ESTscan and HMMER.
We use Perl because it provides modules for many of the typical tasks of bioinformatics programming including:

- Accessing sequence data from local and remote database.
- Transforming formats of database/ file records.
- Manipulating individual sequences.
- Searching for "similar" sequences.
- Creating and manipulating sequence alignments.

Leishmania Major

We decided to use Leishmania genome for our project because it infects people in many countries, including Palestine.

- Leishmania Major characteristics:
    1. The L. major Friedlin genome is 32.8 MB in size.
    2. Including 36 chromosomes.
    3. Number of fragments 776279 KB.
    4. Length of database 360511353 Characters.
    5. Coverage folder ~5.
    6. Average of fragment length 464 Characters.
    7. Size of database 300 MB.
    8. The G+C content is approximately 63%.

2.6.2 Cost-benefit analysis

Experimental analysis was the only way for molecular biologist to determine the nature and location of segmental duplication in the genome of interest. In addition to include laborious procedure, such experimental work would require at least one to three years. In contrast to the tedious experimental work, the proposed bioinformatics system would provide answers for biological questions in very short time. The other advantage of this system over the experimental procedure is the possibility to apply it to any genome that would be sequenced in the future.

Biology Provider benefits:

1. Decrease the biology time and efforts works.
2. Increase the efficiency by increasing the accuracy output.
3. Decreasing the error rate during process.
4. Reducing the cost of biology research.

2.6.3 Economical Study

In this section, we described the resources and costs for the development and implementation requirements.

- **Development Costs:**

Hardware

The following table lists the costs for the hardware that needed to develop this project:

No.	Item	Quantity	Specifications	Cost
1.	Microsoft compatible PC	1	Pentium 4 2400 MHz full cache memory RAM 256 MB Hard Disk drive 40 GHz Floppy drive 1.44 DVD-RW 14X Monitor 17 Keyboard and mouse	$650
2.	Printer	1	Laser hp 1010	$100
Total				**$750**

Table [2.1] Development Hardware Cost

Software:

Table [2.2] shows the software products required and their costs for the system development and implementation.

No.	Software	Cost
1.	Windows XP Service Pack2	$285
2.	Active Perl-5.8.7.813 - MSWin32	Free
3.	Bioperl	Free
4.	BLASTALL	Free
5.	DzSoft.Perl.Editor.v5.6.0.2.WinALL	$49
6.	Microsoft Office2003	$150
Total		**$484**

Table [2.2] Development Software cost

Human Resource cost:

As shown in Table [2.3] the team costs are estimated as the market prices for software developers.

Number	Team Role	Hours/week	Cost\hour	Total
1	Programming	15	$15	$225
2	Software Engineering	10	$15	$150
3	Interface Design GUI	30	$15	$450
Total cost/week				**$825**

Table [2.3] Development Human Resource Cost.

- **Implementation Cost**

This section lists the cost needed to implement this project:

Hardware:

The system can work on a computer Pentium III but it is better to work on a computer which has these specifications.

No.	Item	Quantity	Specifications	Cost
3.	Microsoft compatible PC	1	Pentium 4 3000 MHz full cache memory RAM 1 GB Hard Disk drive 120 GHz Floppy drive 1.44 DVD-RW 14X Monitor 17” Keyboard and mouse	$1010
Total				**$1010**

Table [2.4] Implementation Hardware Cost

Software:

In this section the software needed to implement this project listed:

No.	Software	Cost
1.	Windows Service Back2	$285
2.	Bioperl	Free
3.	BLASTALL	Free
Total		**$285**

Table [2.5] Implementation Software Cost.

- **Other accessories cost:**

Other costs such as books, papers, pens, internet, and transportations are estimated to be $20/ week.

- **Total Cost:**

Table 2.6 below represents the total cost of all system cost:

Number	System Cost	Total
1	Development cost	$2059
2	Implementation cost	$1295
3	Other cost	$300
Total cost		$3654

Table [2.6] Total Cost

2.6.4 Risk Analysis

This section contains the risks that may appear in the project, and the possible solutions:

1. Shortage of project time:

   The time that specified to develop the project is not enough as we need.

2. We have little biology background which would help us developing this system. For that reason we have to lose a lot of time surfing the web and visiting the library.

3. Hardware failure:

   To avoid any failure in the hard ware, and to keep the project, a regular base backup will be made each two days on a different hard disk.

2.6.5 Technical feasibility

This project requires a programming experience in Perl. Team work members have fairly good experience and capabilities to develop such applications. They have experience in different programming languages such as C, Visual Basic, Java, .NET, Perl and others. In addition, we can solve some problems through seeking help from experts people in the university or through bioinformatics forums on the web.

2.6.6 Legal feasibility

Perl is distributed under the GNU's General Public License, which implies that anyone can use, modify, and distribute the source code and documentation of perl. Any modifications made to the source code derived from the GNU-licensed source code must be freely made available to other. This distribution model has encouraged volunteers worldwide to contribute to the perl software.

2.6.7 Time Feasibility

In this section we show how we have allocated the given period over the development stages. The time interval that was available to develop the system was 15 weeks. We have distributed this interval over all of the development process. Table [2.7] shows the time schedule for all development tasks.

- **Time Schedule:**

As shown below in Table [2.7], all system development tasks are distributed over the available fifteen weeks. Some of these tasks were performed in parallel. Figure [2.1] shows the timeline distribution precisely.

Task	Work	Time in weeks
T1	Information gathering and System specification.	2
T2	Software requirement specification.	2
T3	System Design.	6
T4	Coding and implementation.	6
T5	System Testing.	3
T6	System Maintenance.	2
T7	Documentation.	15

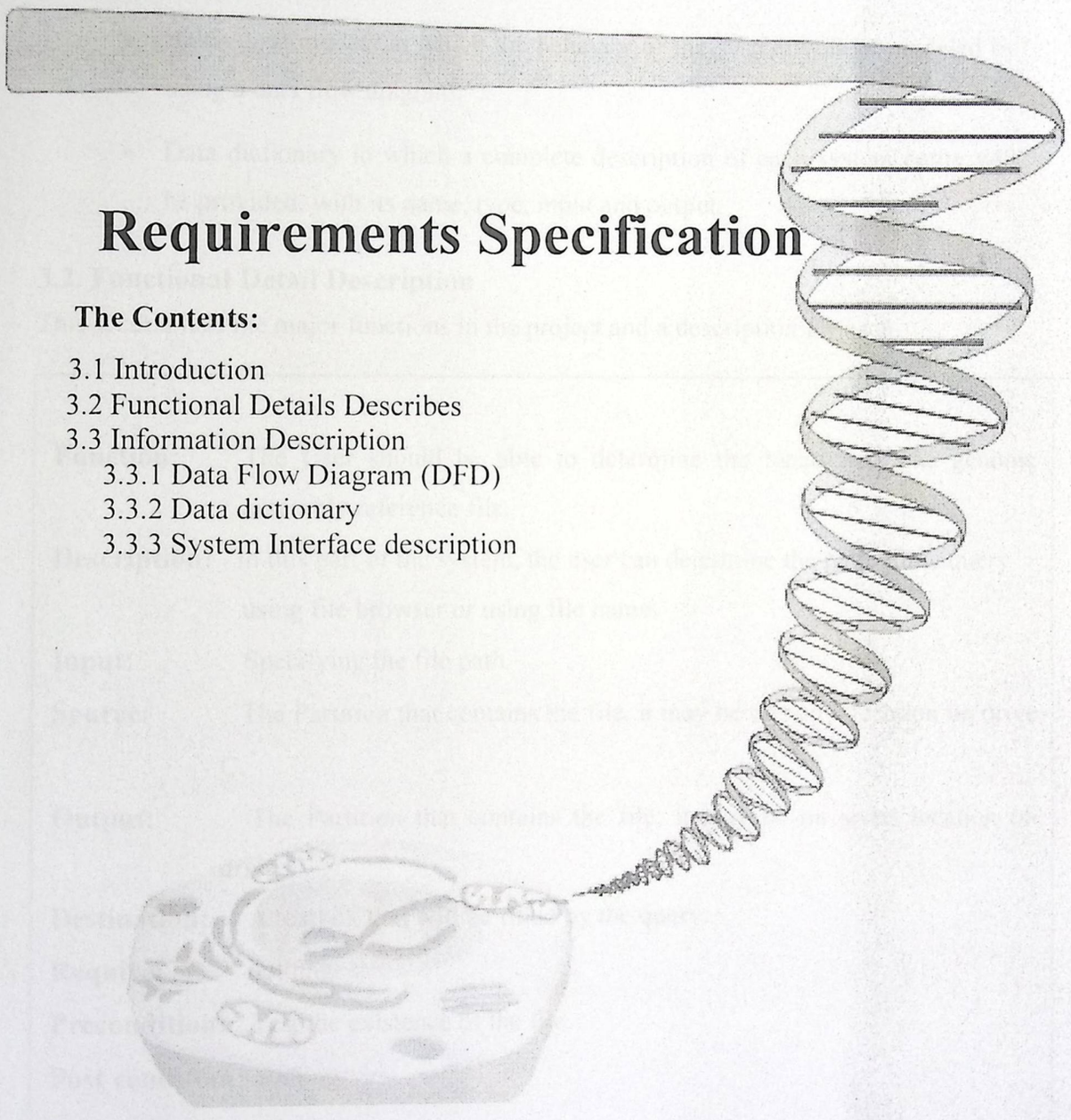
Table [2.7] Time Schedule

- **Gantt chart for time schedule:**

week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2															
3															
4															
5															
6															
7															

Figure [2.1] Gantt chart

3 Chapter Three



Requirements Specification

The Contents:

3.1 Introduction

In this section the software specifications will be addressed and identified in more technical terms, and in more details.

In this section we will cover:

- Functional description of our system, in which all the supported functions and services will be identified and modeled.
- Behavioral models in which the behavior of the system will be modeled by using a data flow diagram.
- Data dictionary in which a complete description of each system entity will be provided, with its name, type, input and output.

3.2. Functional Detail Description

This section lists the major functions in the project and a description for each.

Function: The User should be able to determine the location of the genome assembly reference file.

Description: In this part of the system, the user can determine the path of the query using file browser or using file name.

Input: Specifying the file path.

Source: The Partition that contains the file, it may be on some location on drive C.

Output: The Partition that contains the file; it may be on some location on drive C.

Destination: A textbox that will be filled by the query.

Require: nothing.

Precondition: Test the existence of the file.

Post condition: nothing.

Function: The User should be able to determine the location of the WGSR data file.

Description: In this part of the system, the user can determine the path of the query using file browser or using file name.

Input: Specifying the file path.

Source: The place that contains the file, it may be on some location on drive C.

Output: Place that contains the file; it may be on some location on drive C.

Destination: A textbox that will be filled by the query.

Require: nothing.

Precondition: Test the existence of the file.

Post condition: nothing.

Function: The User should be able to determine the E-value (expectation value).

Description: In this part of the system, the user can determine the E-value.

Input: Specifying the value of E-value.

Source: The place that contains E-value exists in dropdown list on the program interface.

Output: Adjust system parameters

Destination: A dropdown list that will be filled by the user.

Require: nothing.

Precondition: Test the existence of the dropdown list.

Post condition: Nothing.

Function: The User should be able to determine the Incremental value.

Description: In this part of the system, the user can determine the incremental value.

Input: Specifying the value of incremental value.

Source: The place that contains incremental value exists in textbox on the program interface.

Output: Adjust system parameters

Destination: A textbox that will be filled by the user.

Require: nothing.

Precondition: Test the existence of the textbox.

Post condition: Nothing.

Function: Coverage fold.

Description: In this part of the system, the user can determine the number of whole copy gene.

Input: number of coverage fold.

Source: the user of system

Output: Adjust system parameters.

Destination: A textbox that will be filled by number.

Require: nothing.

Precondition: Test the existence of the numeric.

Post condition: nothing.

Function: The User should perform data formatting that is necessary for the subsequent Mega-Blast program

Description: In this part of the system, the systems will change the file to new formatting, to give Mega Blast the ability of reading it.

Input: The file which is selected by the user.

Source: A textbox that will be filled by the Data.

Output: New formatting of the file.

Destination: The input of the MeagBlast.

Require: Determine the location of the genome data file.

Precondition: Test the existence of the Data.

Post condition: Run MegaBlast to get similar EST's.

Function: Run MegaBlast

Description: This program is optimized for aligning sequences that differ slightly as a result of sequencing or other similar

Input: Formatting data file.

Source: Data file.

Output: file results from MegaBlast program which can be run through the system. It contains the headers of the EST`s, the query gene aligned with each EST and information about alignment process.

Destination: To analyzing the MegaBlast output.

Require: Make formatting data.

Precondition: Test the existence of the formatting data.

Post condition: Run MegaBlast to get similar EST's.

Function: Analyze to determine the duplicated segments in the tested genome sequences

Description: Apply the defined criteria to predict if the string contains a duplicated segment.

Input: Output MegaBlast file.

Source: Data file.

Output: Duplication on genome.

Destination: nothing.

Require: MegaBlast Output file.

Precondition: Test the existence of the MegaBlast output file.

Post condition: display the result of anlayze.

3.3 Information Description

This section talks about the data and information in the project, describe the dataflow, and a description for each entities names.

3.3.1 System Data Flow Diagram (DFD)

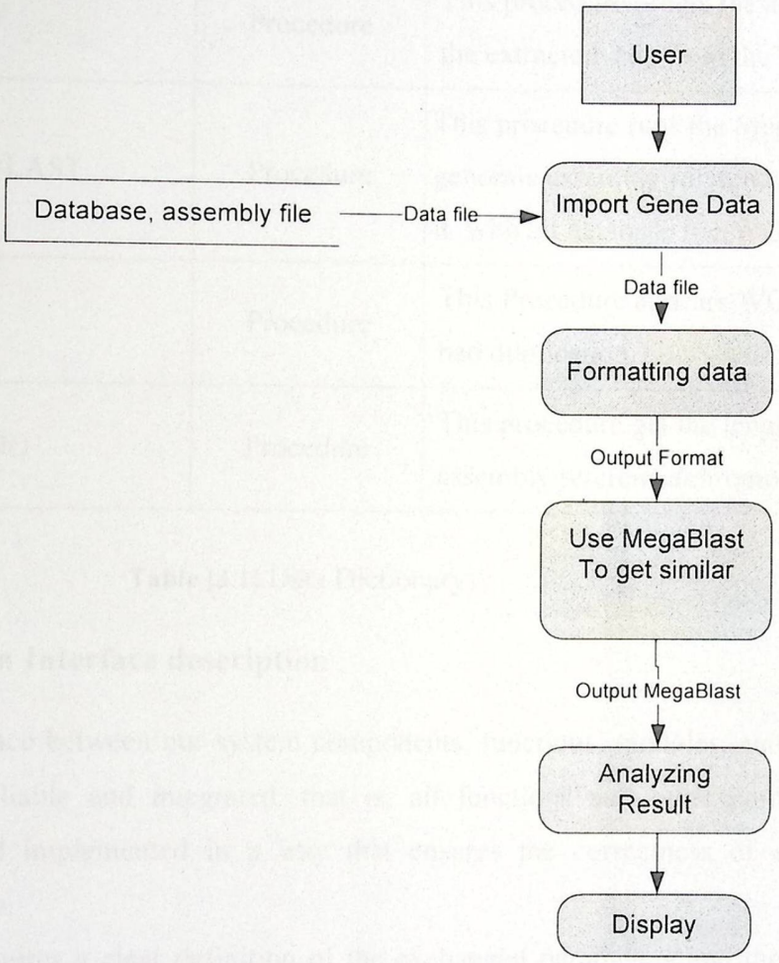
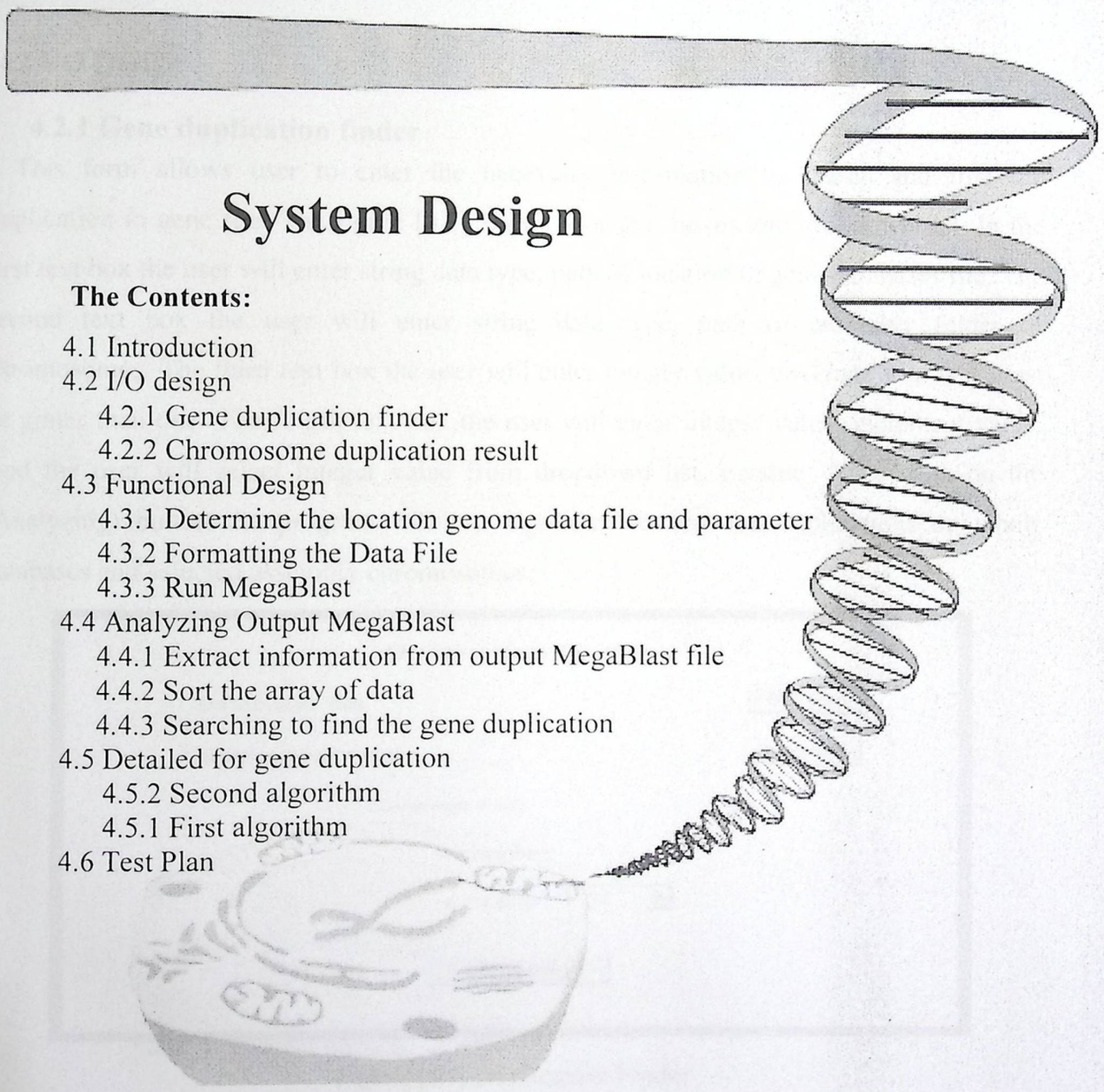


Figure [3.1] System Data Flow

4 Chapter Four



System Design

The Contents:

3.3.2 Data dictionary

Entity name	Type	Description
GET_DATA	Function	This function takes the files that the MegaBlast has carried out then it reads data and extracts the wanted information
SORT ARRAY	Procedure	This procedure orders the array which has the extracted data from the file.
RUN_MEGABLAST	Procedure	This procedure runs the MegaBlast for genomic assembly reference and compares it with all database file(WGSR)
DRAW_DUP	Procedure	This Procedure appears WGSRs which had duplication
GET_LENGTH()	Procedure	This procedure get the length of genomic assembly reference(chromosome file)

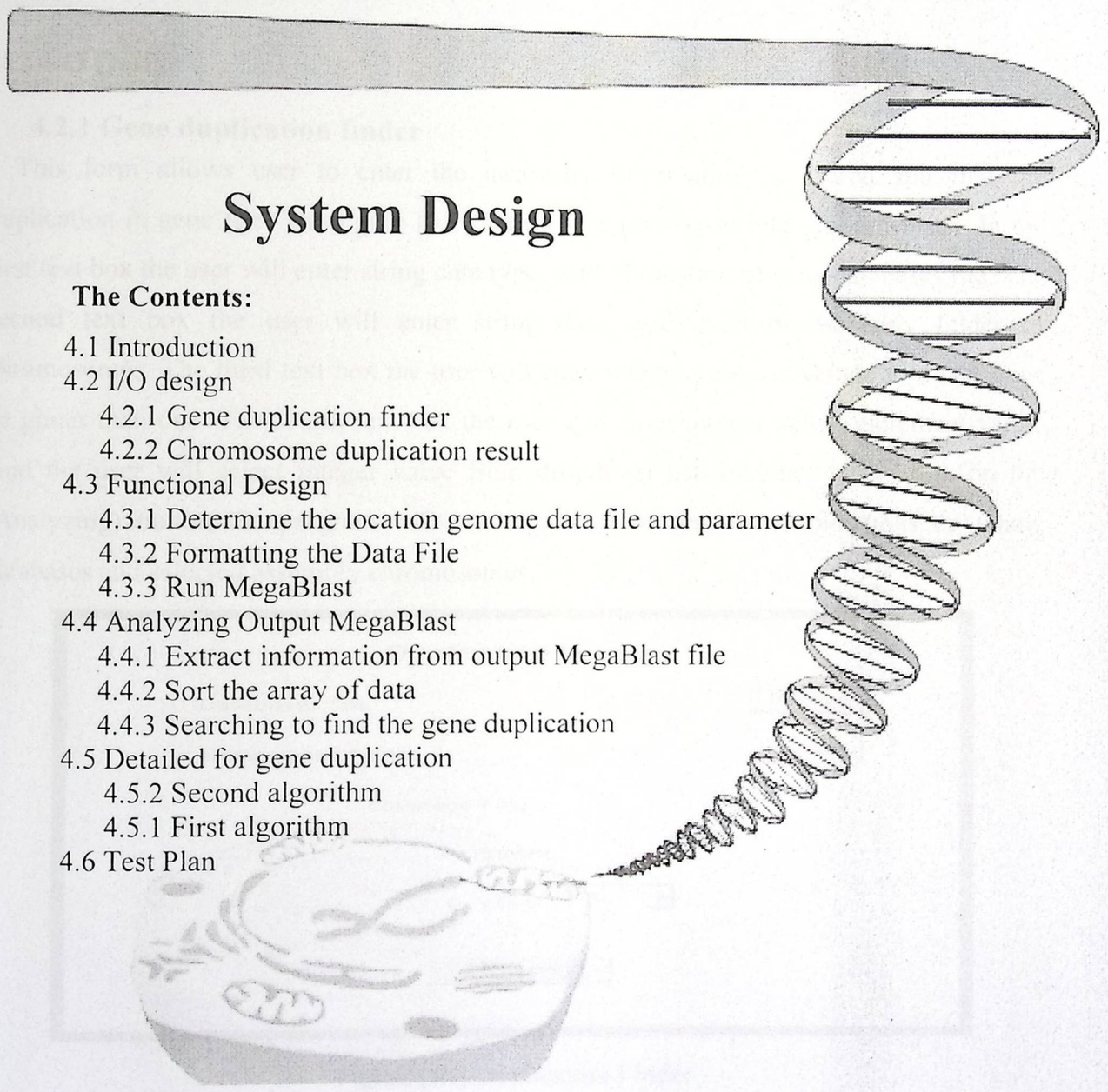
Table [3.1] Data Dictionary

3.3.3 System Interface description

The interface between our system components, functions, modules, and subsystems are to be reliable and integrated, that is, all functions and other components are designed and implemented in a way that ensures the correctness of collaboration between them.

Doing so requires a clear definition of the exchanged parameters and their types and orders, a reliable methodology for dealing with shared memory along with the input/output resources, and other issues that could appear during system design and system testing phases.

4 Chapter Four



System Design

The Contents:

4.1 Introduction

This Chapter describes the System Design, the functional design for all modules in the software system and I/O design.

Topics which covered in this Chapter:

1. I/O Design.
2. Functional design.
3. Test plan.
4. Detailed for gene duplication.

4.2 I/O Design

4.2.1 Gene duplication finder

This form allows user to enter the necessary information to search and find the duplication in gene file. The figure [4.1] shows four text boxes and dropdown list. In the first text box the user will enter string data type, path of location of gene databases file. The second text box the user will enter string data type, path of assembly folder of chromosomes. The third text box the user will enter integer value, coverage fold that must be grater than one. The fourth text box the user will enter integer value, increment value. And the user will select integer value from dropdown list, e-value. After click on the (Analyzing) button the program will run algorithm to find the duplications on whole databases and selected assembly chromosomes.

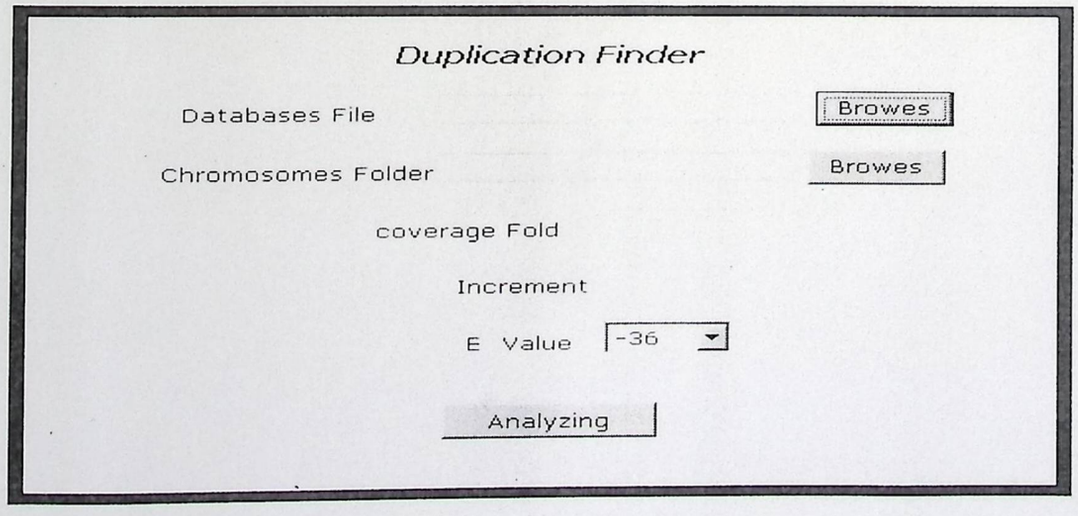


Figure [4.1] Duplication Finder

4.2.2 Chromosomes Duplication Result

This screen displays the result of duplication on the assembly chromosomes that the project found it.

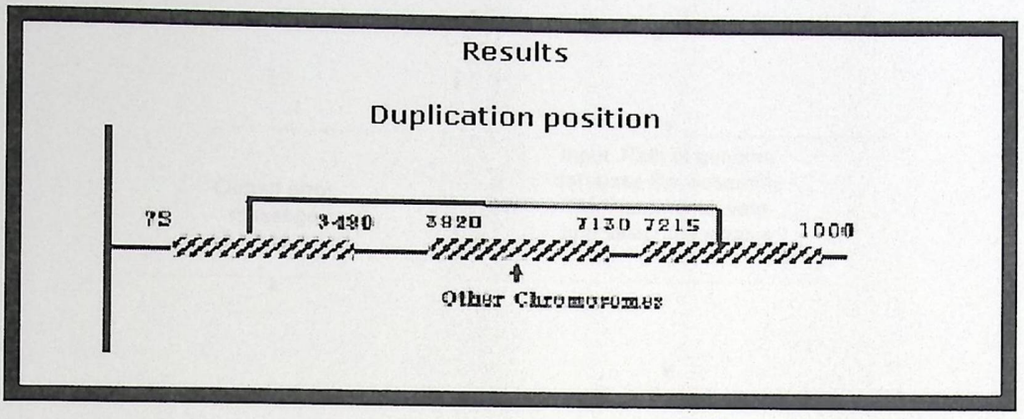


Figure [4. 2] Duplication Result

4.3 Functional Design

This section describes the functional design for each module in the software system.

4.3.1 Determine the location genome data file and parameter

This function describe logical flow to determine the location of database file, assembly file, coverage fold, e-value, and the amount incremental value then we will check the validity of them.

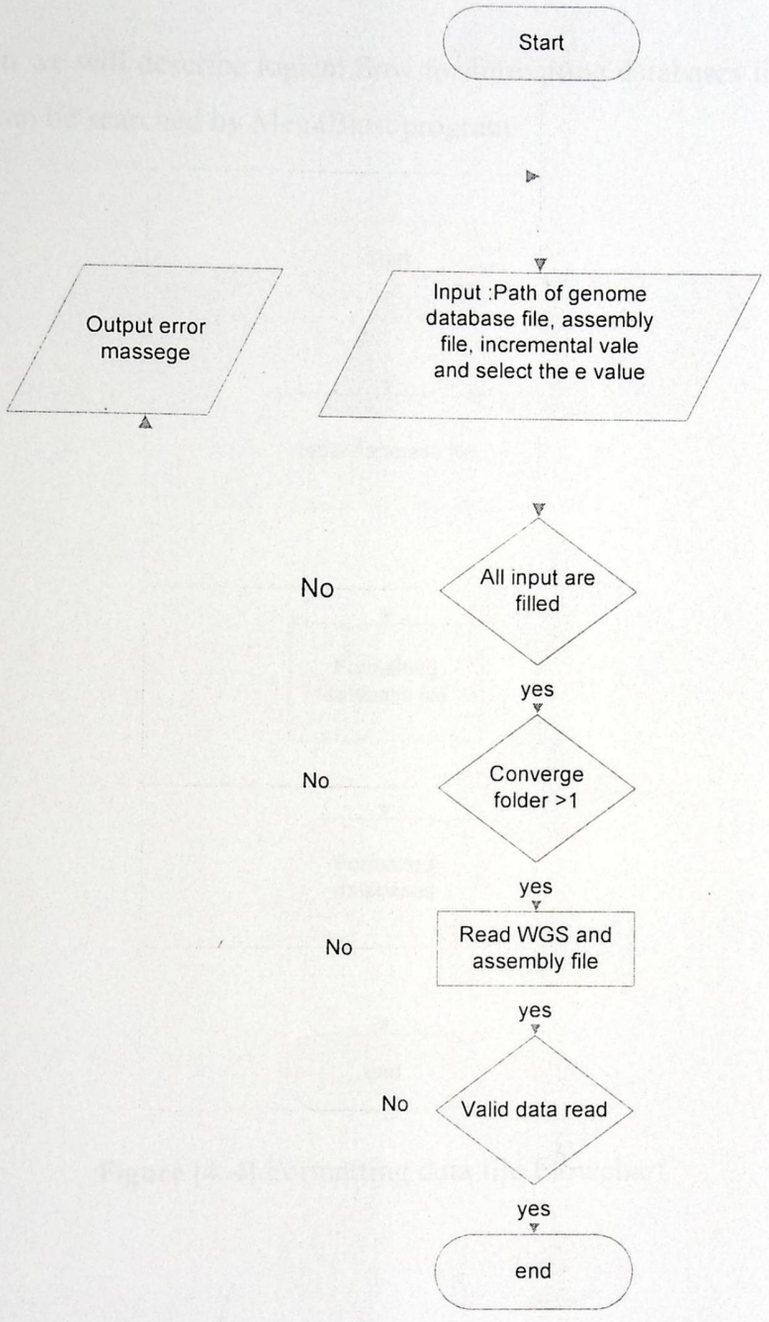


Figure [4. 3] determine the location genome data file and parameter Flowchart

4.3.2 Formatting the Data File

In this function we will describe logical flow for formatting databases file (WGS) before these databases can be searched by MegaBlast program.

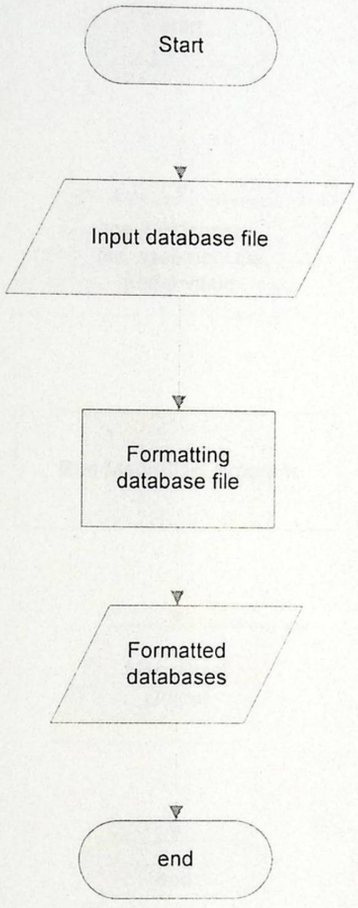


Figure [4. 4] Formatting data file Flowchart

4.3.3 Run MegaBlast

In step, Function will run the MegaBlast to produce optimized for aligning sequences that differ slightly as a result of sequencing or other similar. Execute MegaBlast on databases file with assembly file.

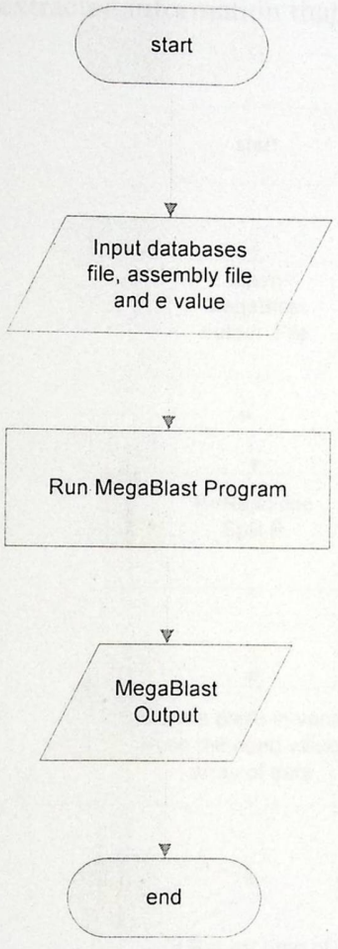


Figure [4. 5] Run MegaBlast

4.4 Analyzing Output MegaBlast

In this function we will describe three functions that are related to it:

4.4.1 Extract information from output MegaBlast file

In this part, we will make processing for the output MegaBlast file which will produce and extracted information that is essential to start searching the gene duplication.

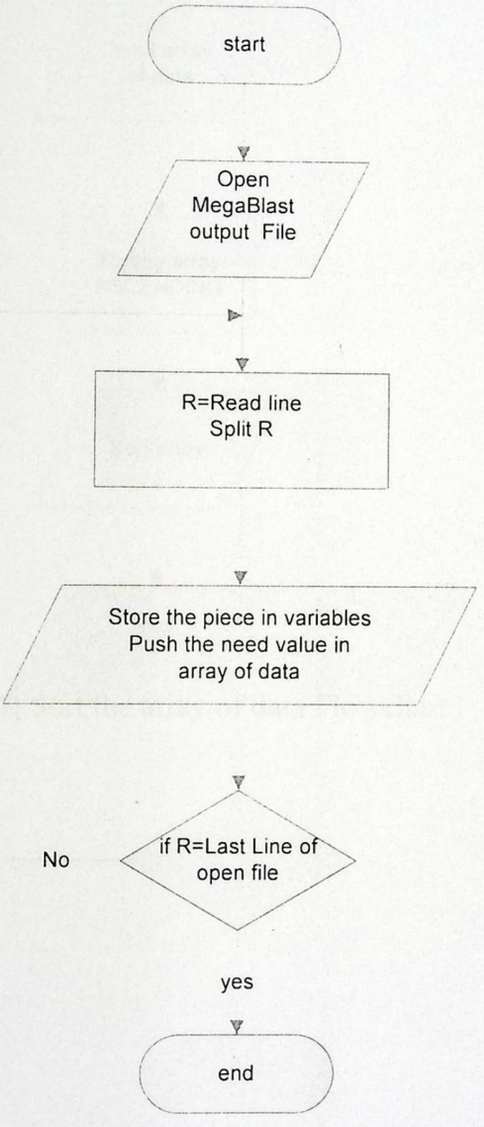


Figure [4. 6] Extract information from output MegaBlast files Flowchart

4.4.2 Sort the array of data

In this part, we will make sorting the array of data ascending according to start of alignment in query using the Bubble Sort algorithms.

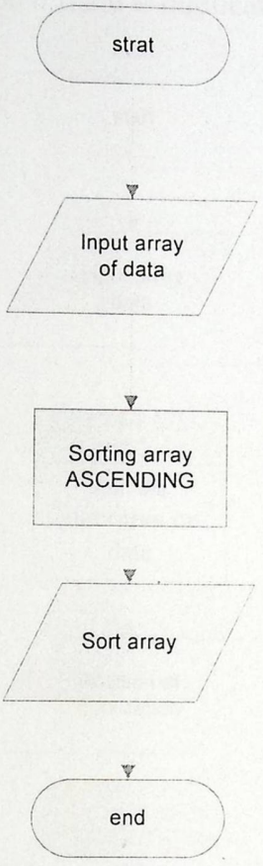


Figure [4. 7] Sort the array of data Flowchart

4.4.3 Searching to find the gene duplication

In this part, we will make processing data that extract from the first step and sort it, and then determine if there is duplication or not.

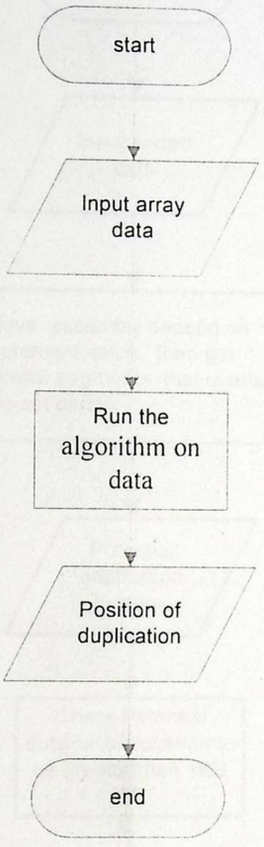


Figure [4. 8] searching to find the gene duplication Flowchart

4.5 Detailed for gene duplication

This section describes the algorithms which we suggested to solve the problem in searching for duplication and to make it understandable used flowchart and pseudocode.

4.5.1 First algorithm

We suggested the first algorithm to find duplication location by checking the position of accumulation in the reference assembly chromosomes, to make this operation easier we cut the assembly to fragments, then the algorithm counts the segments which are in the same

location with fragment and make decision of duplication, but this algorithm has some problems because the large number of short similar segments and because of MegaBlast is Local alignment program.

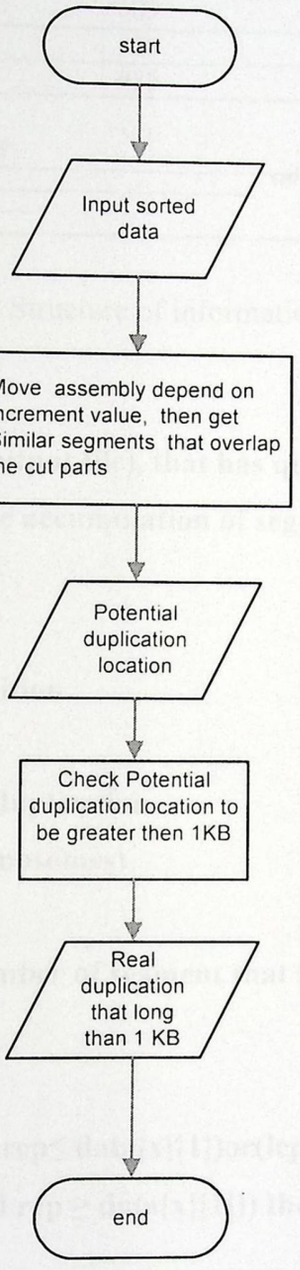


Figure [4. 9] First algorithm

- **Algorithm problems:**

1. Can not determine location of duplication exactly.
2. Gives you expected duplication in some locations but include mistake duplication.
3. Time consumer.

- **Pesdocode of the algorithm:**

start query	End query	length	segments
45	445	400	LM1585Bc08.q1c
96	785	689	LM15Q2c5.p1t
150	1080	930	LM15T3h7.q1t
185	1170	985	LM15L3g8.p1t

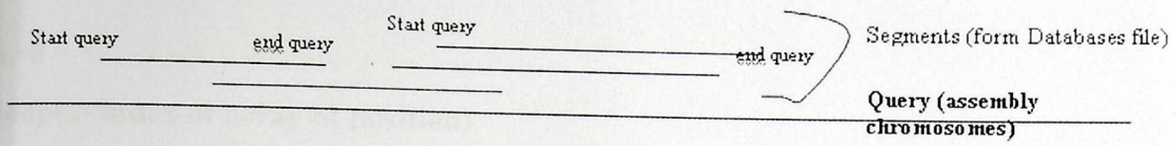


Figure [4. 10] Data Structure of information that analyze it

data[][] #data from MegaBlast(output file), that has query start and end alignment.

Position[x][0] #position that have accumulation of segment will check exactly.

Increment =1000

lcp=0 #initial left check position.

rcp =2000 #initial right check position.

W=0

Pr[][] #location that may be has duplication

loop (right check ≥length of chromosomes)

begin loop

counter=0 # counter the number of segment that location has it.

loop(x<index of array of data)

begin loop

if ((lcp ≥data[x][0] and rcp≤ data[x][1])or(lcp≤data[x][0] and lcp ≥ data[$x][1])or(rcp ≤data[x][0] and rcp ≥ data[x][1])) then

begin if

Counter =Counter +1

end if

x=x +1

end loop

Position[w][0]=counter

```
Position[w[1]= lcp
W=w+1
lcp= lcp + increment
rcp= lrp + increment
end loop

x=0
loop(x<index of array of position)
begin loop
  if(position[x][0]> d #number of accumulations make duplication ) then
     begin if
            per= position[x][0]
     end if
end loop

#check the palace that has may be duplication, to ensure it real duplication, that
length grater than 1KB
cp=0 #check position
c=0 #count the number of segment that position contain
loop(x<index per)
begin loop
    loop(check position< per[x][0]+increment value)
      begin loop
        cp=per[x][0]+1 #depend on the degree of accuracy the biological need)
        c=0;
        loop (x<index data[][])
            begin loop
              if(cp ≤data[$x][0] and cp≥ data[$x][1]) then
               begin if
                dup[d][0]= check position;
                dup[d][1]=data[x][0];
```

```
                dup[d][2]=data[x][1];
                d=d+1;
                c= c+1;
            end if
        end loop
        real duplication[][] =counter
        real duplication[][]= check position;
end loop
```

4.5.2 Second algorithm

To overcome the first algorithm problems we suggested using the Second algorithm, which checks location of duplication by moving in the reference of assembly with specific value, and checks the duplication in that specific value in the assembly chromosome, etc.

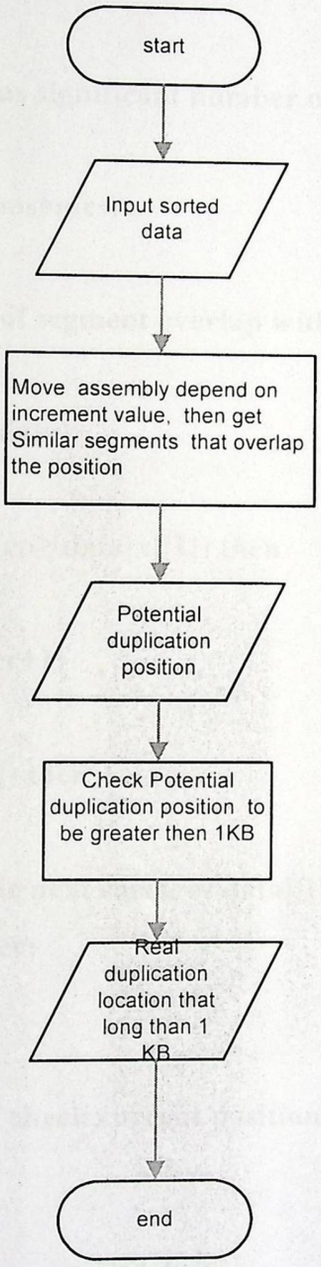


Figure [4. 11] Second algorithm

- **Pesdocode of the algorithm:**

```
Determine the location that may has duplication length grater than 1Kb
data[][]  #data from MegaBlast output file, that has query start and end.
Position[x][0]  #position that have accumulation of segment will check exactly.
Increment =1000 #enter by biology the length of assembly fragmented.
cp=10 #Initial check position
dupx[k][1] #the location that  has significant number of segment

loop1:loop (cp ≥length of chromosomes)
begin loop {loop1}
    Counter=0# count number of segment overlap with check position
     x= start;
      loop (x<index of array of data)
       begin loop
           if(cp ≤data[x][0] and cp≥ data[x][1]) then
               begin if
                 counter =counter+1;
           end if
           else if(cp < data[x][0]) then
            begin if
             start=x # start of the next check of data[][]
             dupx[k][1]= counter;
             dupx[k][0]= cp
             k++;
             next loop; # end of check current position
            end if
             x=x+1
       end loop
cp= cp + increment value
end loop {loop1}
```

```
dupmap[][]# gets position that has significant number of segment
x=0
g=0
loop(x ≥index of dupx)
 begin loop
    if(dupx[x][1]>8) then
     begin if
          dupmap[c][0]=dupx[x][0];
          dupmap[c][1]=dupx[x[1];
          c-=c+1
     end if
    x =x+1
end loop
dupmap[][]# gets position that has significant number of segment
sb=0#start border of duplication
eb=0#end border duplication
dupx1[][]#new sequence position  that has duplication
i=0
loop(i≥index of dupmap) #determent the border of excepted duplication
 begin loop{loop1}
  st=dupmap[i][0]
  e=0 #set the next start loop, that determent if the border sequence or not
  eb =dupmap[i][0]
  j=i+1
     loop(j≥index of dupmap)
    begin loop{loop2}
        if(dupmap[j][0]==dupmap[i][0]+((j-i)* increment value)) then
         begin if
              e=j-i
              end=dupmap[j][0]
         end if
```

```
   end loop{loop2}
   i=i+e;
 if(sb-increment value <0) then
      begin if
         start border =0;
      end if
 else if
       begin if
       start border = start border - increment value;
       end if
if(eb + increment value > length of chromosomes)
    begin if
   eb = len# length of chromosomes
 end if
 else if
    beginif
     eb + increment value;
   end if
 dupp[s][0]= sb #start border
 dupp[s][1]= eb #end border
 s=s+1
 end if

#check if the location is real duplication with length grater than 1KB
dupp][] # location that has significant number of  accumulation.
loop(p≥index of dupp)
 begin loop{loop1}
         y=dupp[p][0]
         label:loop(y≥dupp[p][1])
          begin loop{loop2}
                c=0;
```

```
loop(j≥data)
begin loop{loop3}
 if(y≤data[j][0] and y≥data[j][1]) then
        begin if
        c=c+1
        end if
        else if
         begin if
         if(y<data[j][0]) then
              begin if
              dupx1[k][1]=c;
              dupx1[k][0]=y;
              k=k+1;
              next label  #end of check the current position
              end  if
          j=j+1
    end loop{loop3}
         y=y+1
  end loop{loop2}
 p=p+1
end loop3

dupmapx[][] #get significant position
j=0
loop(j≥index of dupx1)
 begin loop
    if(dupx1[j][1]>converge folder*2 )
          begin if
          dupmapx[g][0]=dupx1[j][0]
          dupmapx[g][1]=dupx1[j][1]
          g=g+1
```

```
                end if
 end loop

dupmapx[][] #significant position, that has duplication
sd =0 # start of duplication
ed=0# end  of duplication

loop(i≥index of dupmapx)
  begin  loop{loop1}
       sd=dupmapx[i][0]
       e=0;
     ed=dupmapx[i][0]
 loop(j≥of dupmapx-1)
   begin loop{loop2}
     if(dupmapx[j][0]==dupmapx[i][0]+((j-i)*5))
      begin if
       e=j-i
       ed=dupmapx[j][0]
      end if
    j=j+1
  end loop{loop2}
     i=i+e;
     duppx[s][0]= sd
     duppx[s][1]= ed
     s=s+1

end loop{loop1}



#get the dupplcation that has length grater than 1KB
```

```
loop(j≥index of duppx)
 begin loop
  if (duppx[j][0]-duppx[j][1]>1000 KB) then
    begin if
     real duplication[R][0] = duppx[j][0]
     real duplication[R][1] = duppx[j][1]
     r increment by one
    end if
end loop
```

4.6 Test Plan

Here we describe the methodology that we have adapted to test the system, steps that will be followed in the system testing are described bellow:

- **Testing steps:**

1. Unit and Module testing:

We will use unit testing to ensure that each function or module will operate as expected. The test components are; import assembly file, import formatted database file, extract data from output MegaBlast results, searching the gene duplication, display duplication gene result, test the algorithm to be optimal.

2. Integration testing and System testing:

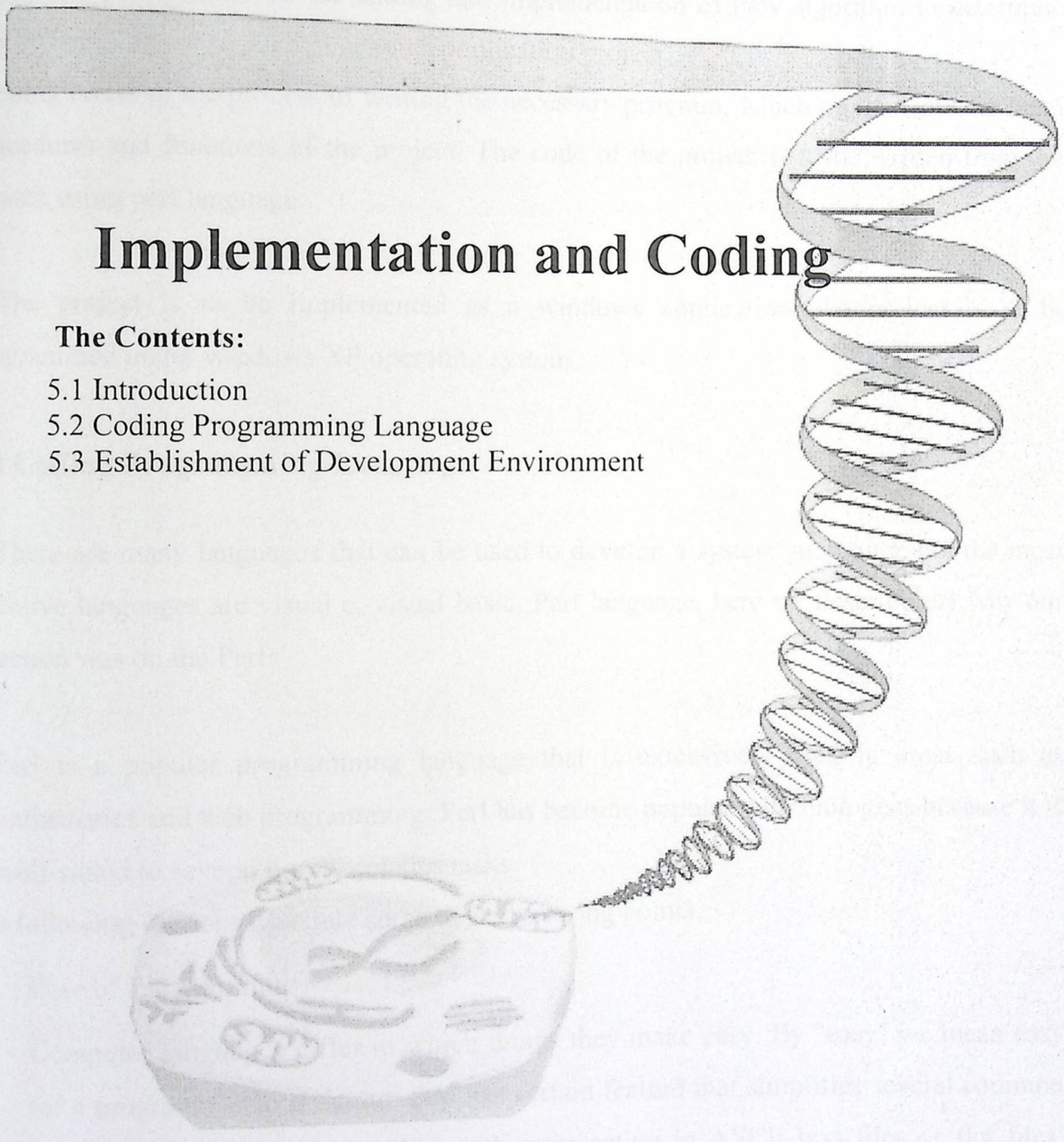
The integration of all units will be tested to ensure that the sub-systems work together properly as it's expected. Sub-systems are integrated to constitute the whole system. System is tested with a complete process of import the assembly file and database, execute the

MegaBlast program, extract data from output MegaBlast file, searching to determine the gene duplication, display the results of gene duplication and try to check the results are real duplication or not.

3. Acceptance Testing:

A system is developing for a single user. The acceptance testing process continues until the system developer and the user agree that the delivered system is an acceptable implementation of the system requirements.

5 Chapter Five



Implementation and Coding

The Contents:

5.1 Introduction

This chapter describes the main steps must be followed to start in coding and programming to reach the design that is described in the previous chapters, and to discover the programming language that is used for this purpose.

This chapter focuses on the coding and implementation of new algorithm to determine highly homologues segmental genome duplication project.
Coding refers to the process of writing the necessary program, which implements the main procedures and functions of the project. The code of the project is to be written from the scratch using perl language.

The project is to be implemented as a windows application, the project is to be programmed under windows XP operating system.

5.2 Coding Programming Language

There are many languages that can be used to develop a system such ours, but the most effective languages are visual c, visual basic, Perl language, here we describe the why our selection was on the Perl:

Perl is a popular programming language that is extensively used in areas such as bioinformatics and web programming. Perl has become popular with biologists because it is so well-suited to several bioinformatics tasks.
The following sections illustrate some of Perl's strong points.

1. Ease of Programming :

   Computer languages differ in which things they make easy. By "easy" we mean easy for a programmer to program. Perl has certain feature that simplifies several common bioinformatics tasks. It can deal with information in ASCII text files or flat files,

which are exactly the kinds of files in which much important biological data appears, in the GenBank and PDB databases, among others.

2. Free Source :

Perl is distributed under the GNU's General Public License, which implies that anyone can use, modify, and distribute the source code and documentation of perl. Any modifications made to the source code derived from the GNU-licensed source code must be freely made available to other. This distribution model has encouraged volunteers worldwide to contribute to the perl software.

3. Rapid Prototyping

Another important benefit of using Perl for biological research is the speed with which a programmer can write a typical Perl program (referred to as rapid prototyping). Many problems can be solved in far fewer lines of Perl code than in C or Java. This has been important to its success in research. In a research environment there are frequent needs for programs that do something new, that are needed only once or occasionally, or that need to be frequently modified. This rapid prototyping ability is often a key consideration when choosing Perl for a job

4. Portability, Speed, and Program Maintenance

Portability means how many types of computer systems the language can run on. Perl has no problems there, as it is available for virtually all modern computers found in biology labs. If you write a DNA analyzer in Perl on your Mac, then move it to a Windows computer, you will find it usually runs as is or with only minor retrofitting.

Speed means the speed with which the program runs. Here Perl is pretty good but not the best. For speed of execution, the usual language of choice is C. A program written in C typically runs two or more times faster than the comparable Perl program. In many organizations, programs are first written in Perl, and then only the programs that absolutely need to have maximum speed are rewritten in C. The fact is maximum speed is only occasionally an important consideration.

Program maintenance is the general activity of keeping everything working such as; adding features to a program, extending it to handle more types of input, porting it to run on other computer systems, fixing bugs, and so forth. Programs take a certain amount of time, effort and cost to write, but successful programs end up costing more to maintain than they did to write in the first place. It is important to write in a language, and in a style, that makes maintenance relatively easy, and Perl allows you to do so.

We have appended the source code written manually for the main functionalities in our application.

5.3 Establishment of Development Environment

- Purchase the computers and the software required for developing the system.
- Install windows XP.
- Install the required utilities.
- Install the Active Perl 5.8.7.813

Before installing the active perl some windows Prerequisites mast be available its as following:

- Hardware: 90 MB hard disk space for typical install.
- Perl for ISAPI: requires an ISAPI-compatible web server, such as IIS 4.0 or greater, or PWS 4.0 or greater.
- Perl Script: requires an ActiveX scripting host such as Internet Explorer 4.0 or greater or Windows Scripting Host.
- Perl Environment Variables: if Perl environment variables such as PERLLIB, PERL5LIB or PERL5OPT have been set on your system, you should unset them before installing Active Perl. Otherwise, these variables may cause incompatible versions of Perl modules to be used during the installation process.

- **Install the active perl**
  - Download the active perl 5.8.7.813 on the internet from website www.activestate.com
  - After download Run ActivePerl-5.8.7.813 .EXE.
  - After determining the location of perl and installing it the following figure will appear.

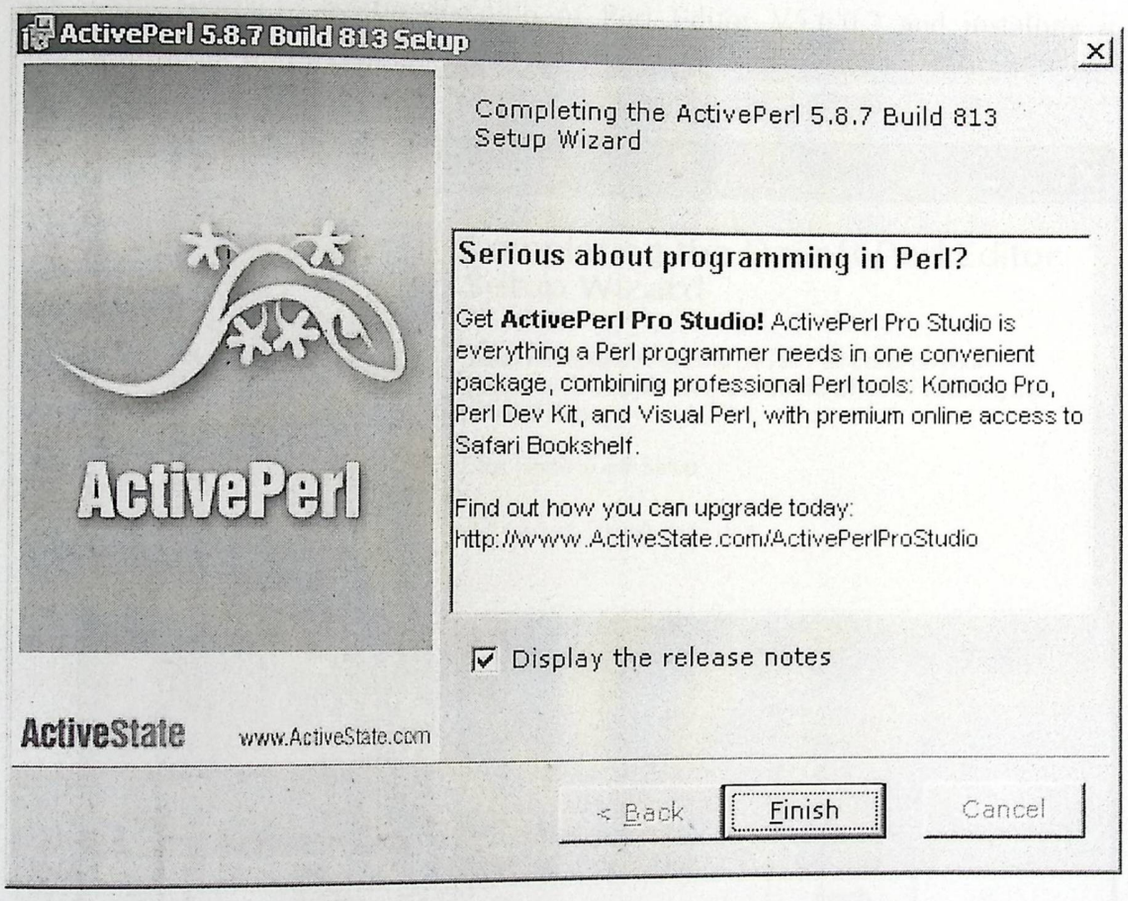


Figure [5.1] Install Active Perl 5.8.7.813.

- **Install DzSoft Perl Editor V5.6.0.2.**

DzSoft Perl Editor is a tool for writing, editing, and debugging Perl CGI scripts. It has a comfortable and intuitive interface both for beginners and advanced programmers. DzSoft Perl Editor is deceptively simple, but it is really a very powerful tool.

To install DzSoft Perl Editor V5.6.0.2:

- Run Perl Editor V5.6.0.2.exe from CD Rom.
- After determining the location of Perl Editor V5.6.0.2 and installing it the following figure will appear.

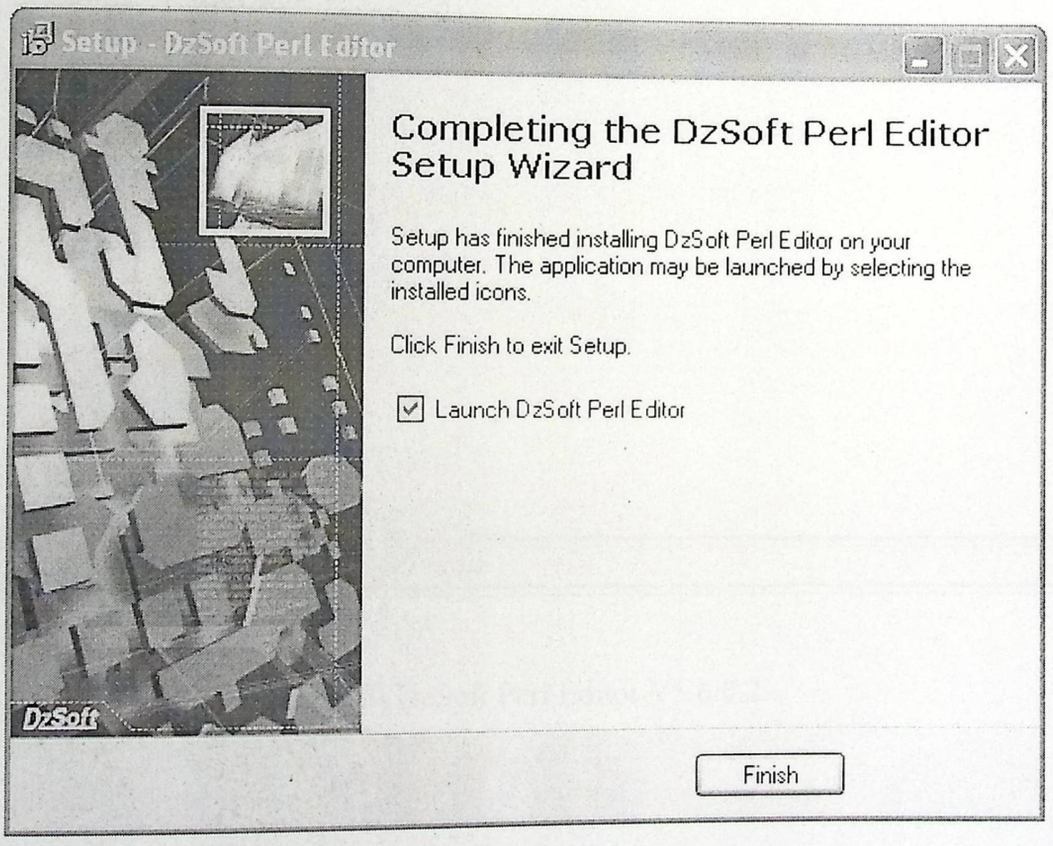


Figure [5.2] Finish installs DzSoft Perl Editor V5.6.0.2.

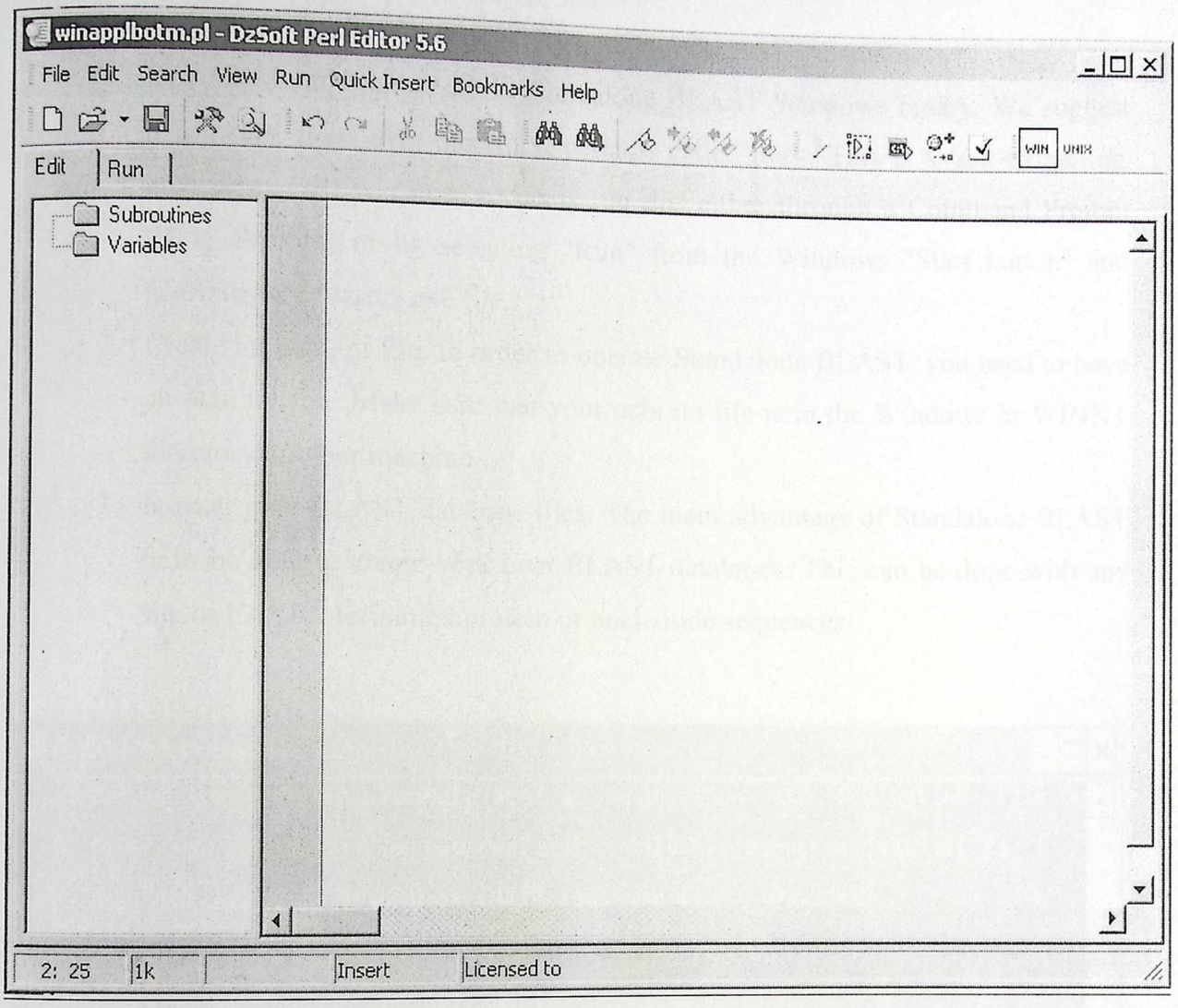


Figure [5.3] DzSoft Perl Editor V5.6.0.2.

- **Install BLAST all Program**

There are three steps needed to setup the Standalone BLAST executable:

1. Download and compress the Standalone BLAST Windows binary. We suggest doing this in its own directory, perhaps called blast. This is a 'self-extracting' archive and all you need to do is run this either through a Command Prompt (DOS Prompt) or by selecting "Run" from the Windows "Start button" and browsing the blastcz.exe file.
2. Create an ncbi.ini file. In order to operate Standalone BLAST, you need to have an ncbi.ini file. Make sure that your ncbi.ini file is in the Windows or WINNT directory on your machine.
3. Format your BLAST database files. The main advantage of Standalone BLAST is to be able to create your own BLAST databases. This can be done with any file of FASTA formatted protein or nucleotide sequences.

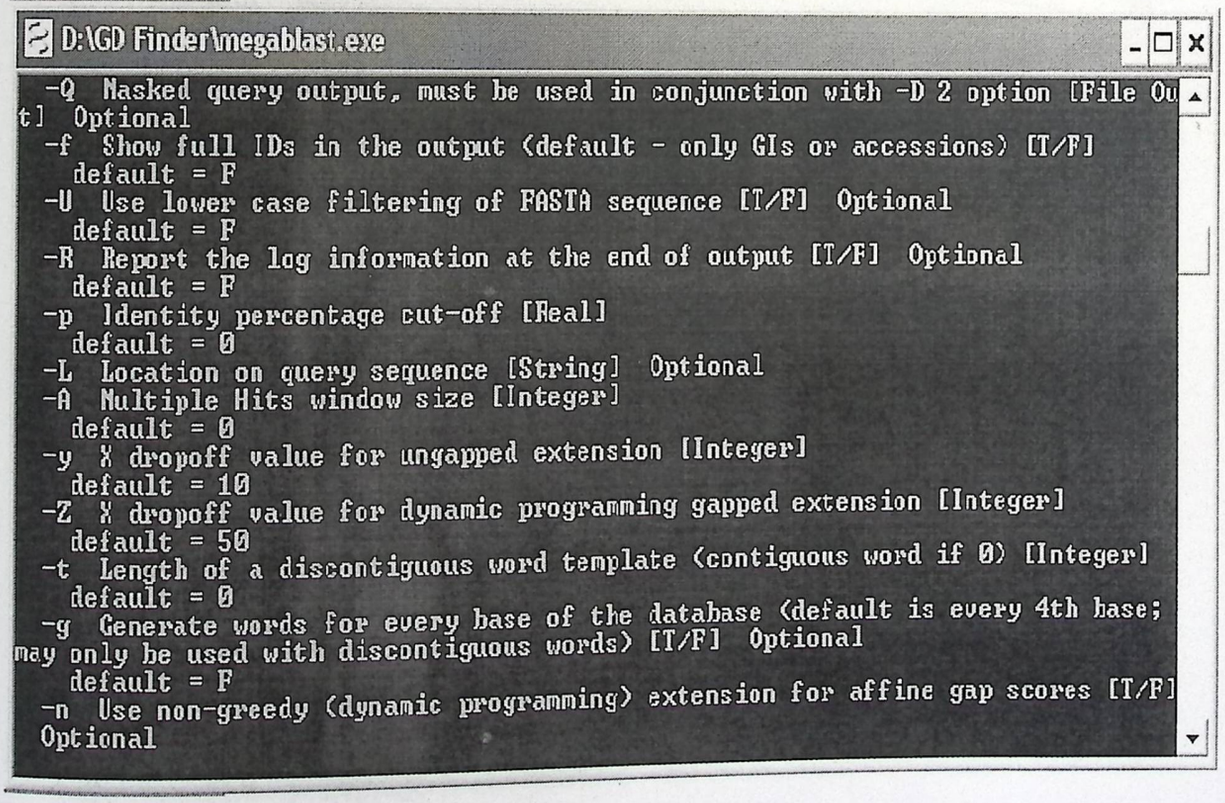
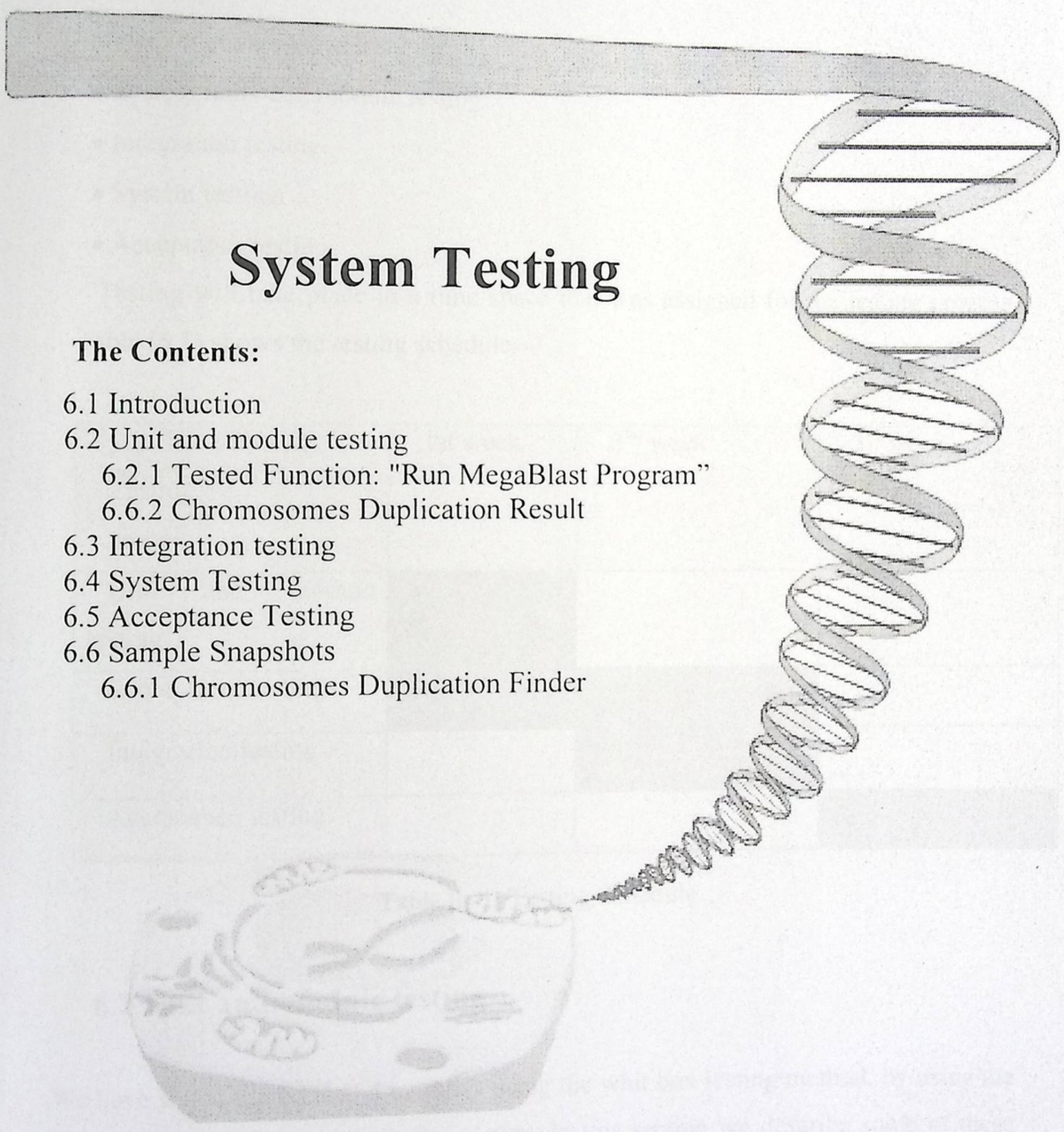


Figure [5.4] install MegaBlast program

6 Chapter Six



System Testing

The Contents:

6.1 Introduction

Testing the system to ensure that it meets its specifications is one of the most important stages in the software system development.

For the purpose of delivering a system that works properly as expected, certain testing procedures should be performed on system and its components; accordingly with an acceptance testing that may be stated as a result for the success of the testing process.

This chapter covers the testing for:

- System units and module testing.
- Integration testing.
- System testing.
- Acceptance testing.

Testing will take place in a time space that was assigned for the testing process. Table (6.1) shows the testing schedule:

Time in week / Testing process	1st week	2nd week	3rd week
Unit and module testing			
Sub-system testing.			
Integration testing.			
Acceptance testing.			

Table [6. 1] Testing Schedule.

6.2 Unit and module testing

We have tested the units and modules using the whit box testing method, by using the path testing on each function in the system. In this section we describe some of these testing procedures on a number of selected functions that are classified as units and

modules, these testing procedures are described here according with a certain snapshots that were captured from the real operating system interface.

6.2.1 Tested Function: "Run MegaBlast Program":

Method: path testing.

Test cases: each test case covers the set of input values in a certain execution path as shown in the function flowchart figure [6.1].

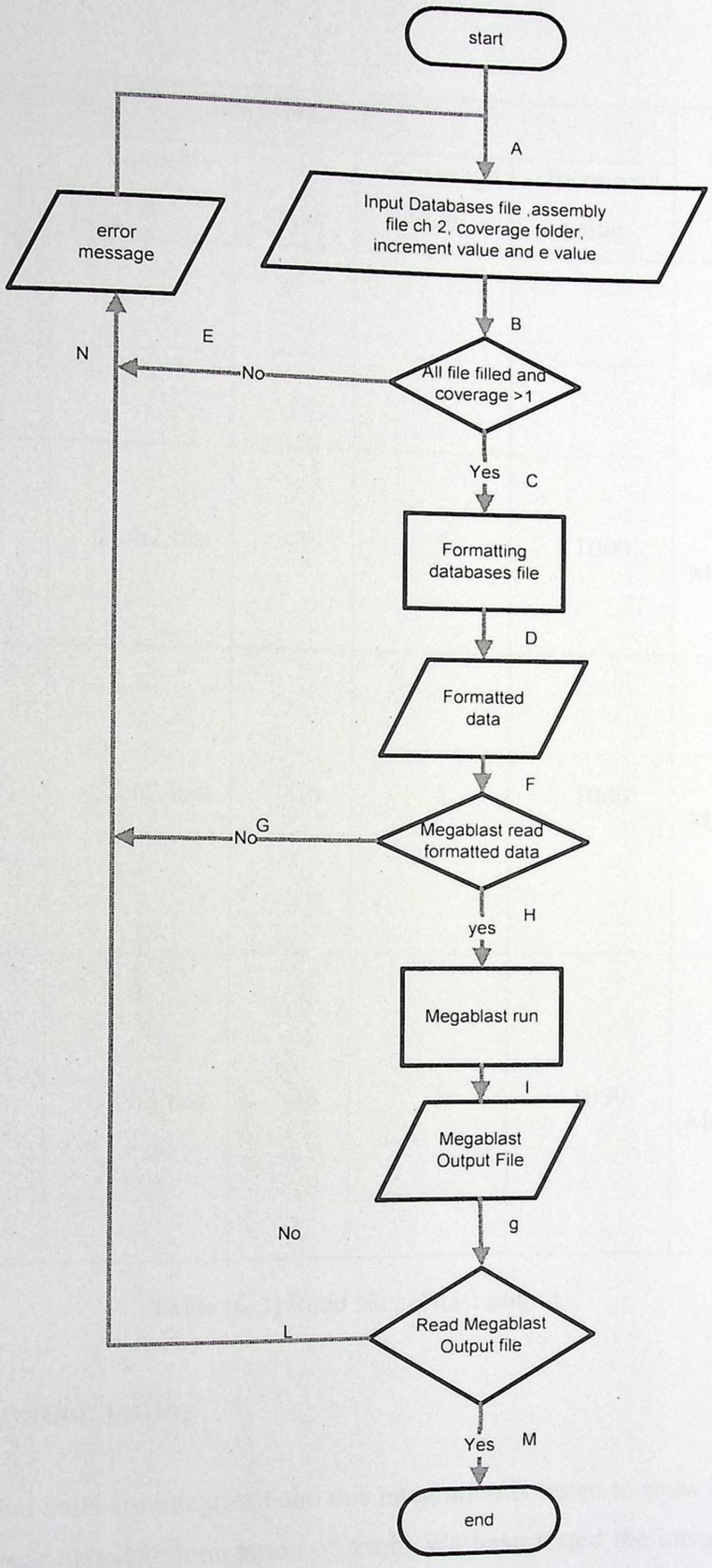


Figure [6. 1] Read MegaBlast output

Test Cases Path	Test Data					Expected Output	Actual Output
	WGS Reads File	Assembly file	E value	Coverage Fold	Increment value		
A-B-E-N	_____	_____	____	___	_____	Run MegaBlast	Not valid MegaBlast output
A-B-C-D-F-G-N	C:\data.file	C:ch2.fast	-36	5	1000	Run MegaBlast	Not valid formatting file
A-B-C-D-F-G-H-I-G-L-N	C:\data.file	C:ch2.fast	-36	5	1000	Run MegaBlast	Not valid MegaBlast output file
A-B-C-D-F-G-H-I-G-M	C:\data.file	C:ch2.fast	-36	5	1000	Run MegaBlast	Run MegaBlast

Table [6. 2] Read MegaBlast output

6.3 Integration testing

All module, and units are integrated and this integration is tested to show if there were defects that appear upon the integration of them. We have tested the integration using top-down testing. Testing here demonstrates on the interfaces between all modules, and the functionality of the integrated parts.

After testing the integration of all subsystems, the result indicated that they work together properly.

6.4 System Testing

The system was tested under several conditions, some errors were detected, and upon these results, we have solved these problems and we imposed the system another time to testing techniques to ensure that it disposed all types of defects and problems.

6.5 Acceptance Testing

The system was tested against its requirements, we conclude that it achieves its functional requirements, and could operate soon in the real environment.

6.6 Sample Snapshots

We have selected some program snapshots to be displayed here to show how the real program behaves when working under certain situations and these snapshots are describe the main functions of our system as shown bellow:

6.6.1 Chromosomes Duplication Finder

This form allows user to enter the necessary information to search and to find the duplication on gene file. The figure [6.2] shows four text boxes and dropdown list. In the first text box the user will enter string value, path of location of gene databases file. The second text box the user will enter string value, path of assembly folder of chromosomes. The third text box the user will enter integer value, coverage fold that must be grater than two. The fourth text box the user will enter integer value, increment value. And the user will select integer value from dropdown list, e value. After click on the (Analyzing) button the program will run algorithm to find the duplications on whole databases and all assembly chromosomes.

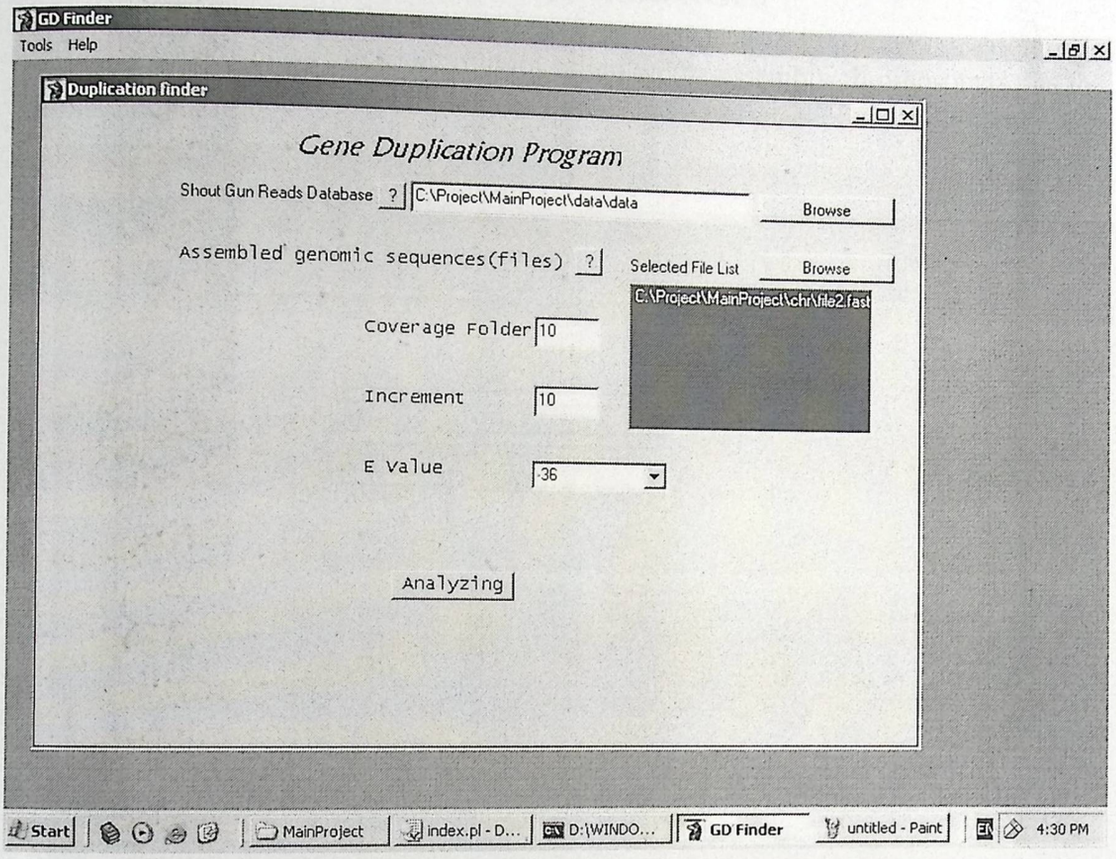


Figure [6. 2] Chromosomes Duplication Finder

6.6.2 Chromosomes Duplication Result

This screen displays the result of duplication on the assembly chromosomes that the project found it.

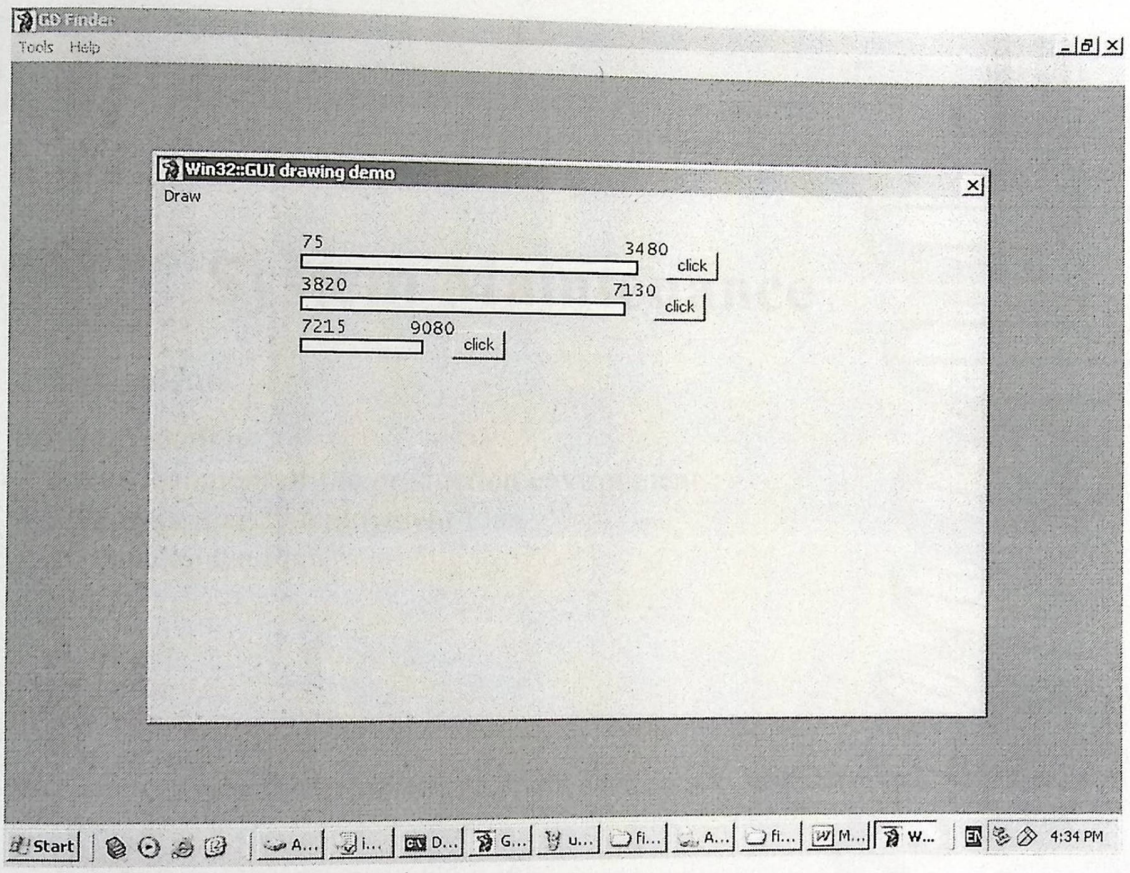
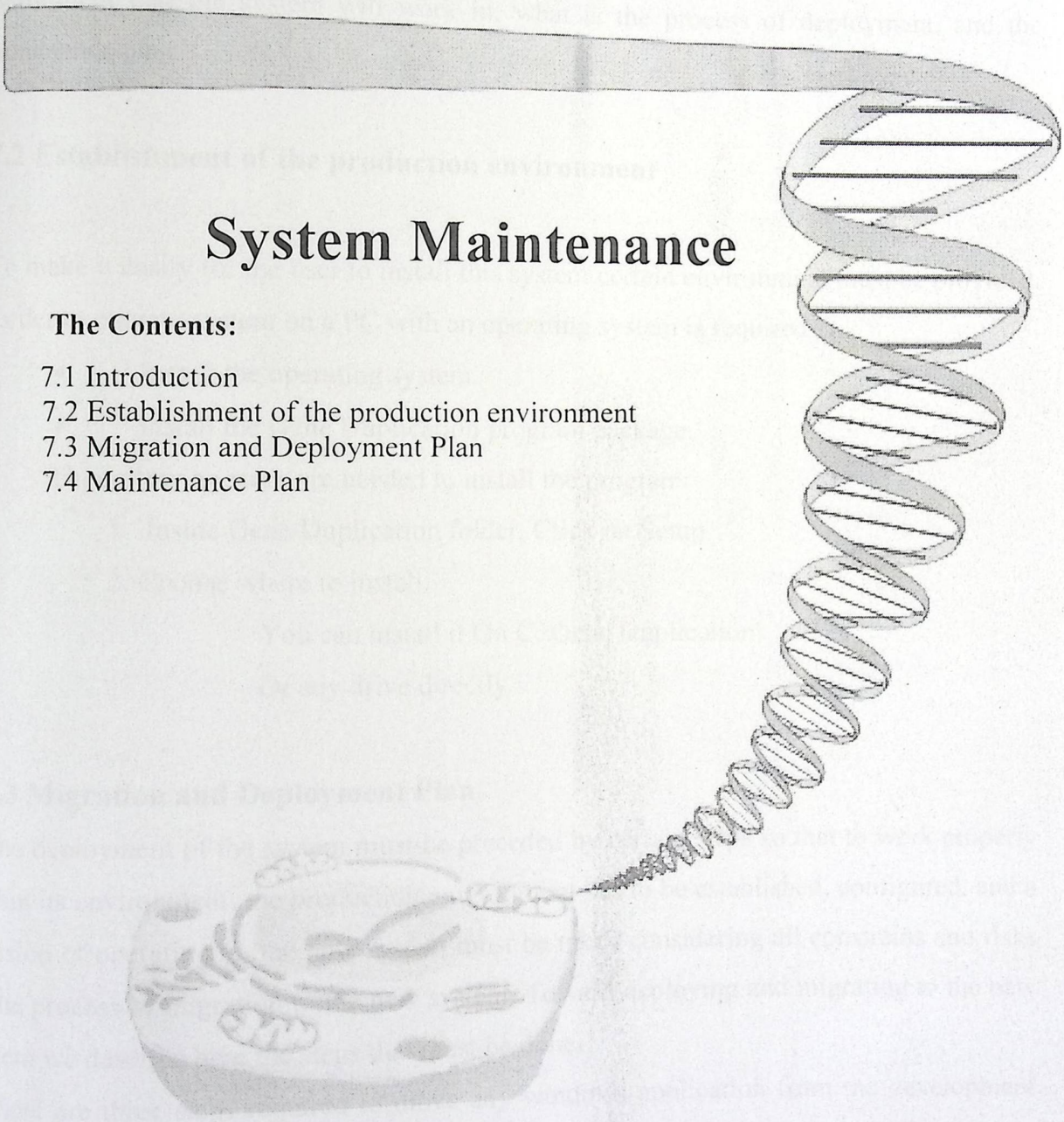


Figure [6. 3] Chromosomes Duplication Result

7 Chapter Seven



System Maintenance

The Contents:

7.1 Introduction

At this chapter we will provide and explain the process and techniques that guidelines the system developer to keep tracing and maintaining the system after running it.

We will talk about the working environment. It is silly to think that the user is the developer of the system because he is just a user who lacks the knowledge of this system in that case, we have to provide the user with the sufficient information about the system, how it works, and how it could be maintained.

This chapter describes how to start working with the system; the establishment of the environment that the system will work in, what is the process of deployment, and the maintenance plan.

7.2 Establishment of the production environment

To make it easily for the user to install this system certain environment must be provided. In order to operate system on a PC with an operating system is required:

- Install the operating system.
- Install the Gene Duplication program package.

The following steps are needed to install the program:

1. Inside Gene Duplication folder, Click on Setup
2. Choose where to install.

   You can install it On C:\Gene Duplication\

   Or any drive directly.

7.3 Migration and Deployment Plan

The deployment of the system must be preceded by certain steps so that to work properly within its environment; the production environment has to be established, configured, and a decision of operating on the new system must be taken considering all constrains and risks of the process of migration to the new system. Toward deploying and migrating to the new system we describe here the steps that must be done.

There are three steps required to move any windows application from the development environment to a production directory:

1. Build, or compile the Windows application, this compilation creates an .EXE file in the directory that contains the code for Gene Duplication.

2. Copy the necessary windows application files(after compilation Gene Duplication) in the development directory to the production directory, which are:
   - The .PL files.
   - Megablast.exe and Formatdb.exe
   - The Execution File
3. Package and build the production solution and run.

7.4 Maintenance Plan

When running system failures or errors may occur. In this case the programmer can follow the code and maintain it, a single error or multiple errors may occur will be solved. If an error handling occurs during the implementation, the error message and a description of that error will be displayed on the screen, then the customer must call the vendor and tell him about the error.

When the programmer solves the error, he should make unit testing and integration testing to ensure that the last qualifications will not influence the whole system performance. Finally, the corrected components must be developed and published.

The program purchaser can call the programmer using a special form, which contains the error information and description, and then the programmer must record each step in another form.

The program purchaser can call the programmer using a special form (Fig 6.1), which contains the error information and description, and then the programmer must record each step in another form (Fig 7.1).

Software Change Request Form

Software Change Request (SCR)

REQUESTED BY ---------- DATE ----------

DEPARTMENT ----------

LOCATION----------

Type

[ ] New Requirement [ ] System Problem [ ] Suggestion for Improvement

[ ] Requirement Change [ ] User Interface Problem [ ] Other:_____

[ ] Design Change [ ] Documentation Correction

Problem description:

_____

_____

_____

_____

Please attach supporting documentation for the requested change
(Screen/report printouts, document pages affected, etc.)

Figure [7.1] Software Change Request Form

References

1. http://www.perl.com/pub/q/documentation.
2. http://www.sanger.ac.uk.
3. http://www.who.int.
4. http://perl-win32-gui.sourceforge.net/cgi-bin/docs.cgi.
5. http://www.NCBI.nlm.nih.gov/blast.
6. Sharabati Talat, **Novel algorithm to determine highly homologous segmental genome duplication**. PPU. 2006 Jan.
7. Jambeck, Per and Gibas, Cynthia. **Developing Bioinformatics Computer Skills.** 1ST Edition O'Reilly & Associates.2001.
8. Lesk, Arthur. **Introduction to Bioinformatics. Oxford University Press**. 1ST Edition 2002.
9. Eichler EE. **Recent duplication, domain accretion and the dynamic mutation of the human genome**. Trends Genet. 2001 Nov;17(11):661-9.
10. Bailey JA, Eichler EE. **Genome-wide detection and analysis of recent segmental duplications within mammalian organisms**. Cold Spring Harb Symp Quant Biol. 2003;68:115-24.
11. Bailey JA, Yavor AM, Massa HF, Trask BJ, Eichler EE. **Segmental duplications: organization and impact within the current human genome project assembly**. Genome Res. 2001 Jun;11(6):1005-17.
12. Bailey JA, Gu Z, Clark RA, Reinert K, Samonte RV, Schwartz S, Adams MD, Myers EW, Li PW, Eichler EE. **Recent segmental duplications in the human genome. Science.** 2002 Aug 9;297(5583):1003-7.
13. Brown, T. A. **Genomes**. BIOS Scientific publisher. Oxford, UK 2nd Edition 2002.
14. Sommerville, Ian, ***Software Engineering***, 7th edition, Addison-Wesley, 2001.
15. Lesk, Arthur. **Introduction to Bioinformatics**. Oxford University Press. 1ST Edition 2002.

16. Jambeck, Per and Gibas, Cynthia. **Developing Bioinformatics Computer Skills.** First edition O'Reilly & Associates.2001. United state of America.

17. Tisdall, James. **Beginning Perl for Bioinformatics**. First edition O'Reilly & Associates.2001. United state of America.

Appendixes

Appendixes

```
# Gene Duplication program designed to analyze real databases
#that contain hug number of shotgun reads that are produced by
#genome projects
use win32;
use Win32::GUI;
 ## engage the Win32::gui module
 my $DOS = Win32::GUI::GetPerlWindow();
Win32::GUI::Hide($DOS);
$ChildCount = 0;
        $o=1;

#################################################################
# Create Main menu.
$Menu = Win32::GUI::MakeMenu(
    "&Tools"                                => "Tools",
    "   > &DFinder"                            => { -name => "DFinder",  -
onClick => \&finder },
    "   > -"                                => 0,
   "   > &Draw"                             => { -name => "draw",  -onClick =>
\&draw  },
    "   > &format data"                                => { -name => "format",  -
onClick => \&format  },
    "   > E&xit"                            => { -name => "File_Exit", -onClick
=> sub {Win32::GUI::Show($DOS);exit(0);} },
    "&Help"                                 => "Help",
    "   > &Tutorial"                          => { -name => "About", -onClick =>
sub { system(" c://project//mainproject//a.swf");} },
     "   > &About"                          => { -name => "About1", -onClick
=>\&about },
);
#################################################################
# First we create an MDIFrame window.
$Window = new Win32::GUI::MDIFrame (
    -title  => "Duplication Finder",
    -left   => 100,
    -top    => 100,
    -width  => 600,
    -height => 400,
    -name   => "Window",
    -menu   => $Menu,
) or die "Window";
#################################################################
# We add an MDIClient window, This window manage Child Window.
$Window->AddMDIClient(
     -name       => "Client",
     -firstchild => 100,                                  # Define Child ID for
menu item
     -windowmenu => $Menu->{Window}->{-handle},  # Define Menu Handle
where Add Window Child name
  ) or die "Client";

#################################################################
# Show main window and go to event loop
$Window->Show;
$Window->Maximize();
 Win32::GUI::ChangeIcon($Window,$icon);
  about();
```

```
Win32::GUI::Dialog();
sub Win_Terminate {
    return -1;
}
###################################################################
# drao location of duplaction
sub draw{
 @dataD;

$Men = Win32::GUI::MakeMenu(
    "&Draw" => "&Draw",
     ">  &draw"    => { -name => "Draw",-onClick => \&Draw_dup },

);

 $Win = new Win32::GUI::Window(

    -left    => 0,
    -top     => 40,
    -name    => "Window",
    -text    => "Duplication finder",
    -menu    => $Men,
    -onTerminate  => sub { $Men->{Draw}->Enabled(1);},
    -width  => 600,
    -height => 400,
    -addstyle     => 1024 | WS_BORDER | WS_CAPTION | WS_SYSMENU,
    );




$Win->Show();
 Win32::GUI::Dialog();

}
# This function create a new child window.
sub finder{

 $font_title = Win32::GUI::Font->new(
                -name => "Lucida Console",
                -size => 15,
                -italic => 1
                -bold =>1 ,
                  );


     $Menu->{DFinder}->Enabled(0);
 $font = Win32::GUI::Font->new(
                -size => 11,
                -bold =>1,
                -quality=>15 );

 $main= $Window->{Client}->AddMDIChild (
        -name => "main" ,
        -text => "Duplication finder" ,
        -sizable => 0,
```

```
        -controlparent =>1,
        -maximizebox =>1,
        -onTerminate  => sub { $Menu->{DFinder}->Enabled(1);},
        -addexstyle =>WS_EX_STATICEDGE,
        -top => 10,
        -left=> 20,
        -width => 640,
        -height =>480,
                        );


 $label1 = $main->AddLabel
  (
  -name=> "label1" ,
  -text=> "Duplication Finder " ,
  -left=>($main->Width()-280)/2,
  -top=> 15,
  -font=>$font_title,
  );

$help1=$main->AddButton
(
  -text=> "?",
  -left=>258,
  -top=>52,
  -width=>20,
  -onClick=> sub {  my $help = new Win32::GUI::Window(
    -parent =>$main,
    -name   => "Window",
    -text   => "help",
    -left     => 250,
    -top      => 140,
    -width  => 200,
    -height => 150,
    -dialogui => 0,
    -addstyle    =>  WS_BORDER | WS_CAPTION | WS_SYSMENU,
    -onTerminate  => sub {return 1;},
);
################################################
#label that comtain help text
              $label2 =$help->AddLabel
             (
             -name=> "label2" ,
             -text=>" Shout Gun Read database should be".
             " a single file containing DNA sequence in fasta".
             " formats that represent the whole genome ".
             " shut gun reads ",
             -left=>10,
             -top=> 10,
              -wrap      => 1,
             -font=>new Win32::GUI::Font(
                         -italic => 1 ,
                         -underline => 1 ,
                         -strikeout => 1,),
             -width=>200,
             -height=>150,
```

```
                                );

$help->Show();
},
  -height=>20,
 );

 $label2 = $main->AddLabel
(
-name=> "label2" ,
-text=> "Shout Gun Reads Database" ,
-left=>62,
-top=> 55,
-font=>$font,
-height=>40,
              );
#####################################################
$text1 = $main->AddTextfield
   (
    -name=> "text1" ,
    -text=> "",
    -left=>280,
    -top=> 50,
    -width=>230,
    -height=>25,
   );
$button1=$main->AddButton
(
  -name=> "open1",
  -text=> "Browse",
  -left=>515,
  -top=>52,
  -width=>100,
  -height=>20,
 );
$label3 = $main->AddLabel
(
-name=> "label3" ,
-text=> "Assembled genomic sequences(files)" ,
-left=>100,
-top=> 100,
-font=>$font,

-height=>40,
);
 $text2 = $main->AddLabel
   (
    -name=> "text2" ,
    -text=> "Selected File List",
    -left=>420,
    -top=> 100,

   );
#*******************************
   $help1=$main->AddButton
    (
        -text=> "?",
```

```
        -left=>380,
        -top=>95,
        -width=>20,
        -onClick=> sub {  my $help = new Win32::GUI::Window(
                    -parent =>$main,
                    -text   => "help",
                    -left     => 250,
                    -top      => 140,
                    -width  => 220,
                    -height => 250,
                    -dialogui => 0,
                    -addstyle    =>  WS_BORDER | WS_CAPTION | WS_SYSMENU,
                    -onTerminate  => sub {return 1;},
                );
#####################################################
#label that comtain help text
              $lab =$help->AddLabel
             (

             -text=>"These are the assembleled chromosome".
             " of the genome of interest. They can be either".
             " a single file or multiple chromosomes. if the genome".
              " of interest has only one chromosomes you can select".
              " the file which contains the DNA sequence sequence of".
              " that specific chromosome. In case you genome of
interest".
              " has multiple chromosomes ,select the file and press
open",
             -left=>5,
             -top=> 5,
              -wrap      => 1,
              -align     => left,
             -font=>new Win32::GUI::Font(
                         -italic => 1 ,
                         -underline => 1 ,
                         -strikeout => 1,),
            -width=>200,
            -height=>250,
                          );

##########################################

             $help->Show();},
            -height=>20,
            );
 #************************
$List1 = $main->AddListbox(
   -name => "List1",

   -left => 420,
   -top  => 120,
   -height => 120,
   -menu => 1,
   -tabstop => 1,
   -group => 1,
   -width => 180, -multisel => 0,
   -addstyle => WS_VSCROLL | WS_VISIBLE | WS_CHILD,
```

```
    -foreground => [255, 255, 255],
    -background => [64, 64, 64],
) or print_and_die("new Listbox");

$button2=$main->AddButton
(
  -name=> "open2",
  -text=> "Browse",
  -left=>515,
  -top=>95,
  -width=>100,
  -height=>20,
 );
$label4= $main->AddLabel
(
-name=> "label4" ,
-text=> "Coverage Fold " ,
-left=>($main->Width()-180)/2,
-top=> 150,
-font=>$font,
-height=>40,
);
$text3 = $main->AddTextfield
   (
    -name=> "text3" ,
    -text=> "",
    -left=>350,
    -number=> 1,
    -top=> 145,
    -width=>50,
    -height=>25,
   );
$label5= $main->AddLabel
(
-name=> "label5" ,
-text=> "Increment" ,
-left=>270,
-top=> 200,
-font=>$font,
-height=>40,
);
$text4 = $main->AddTextfield
   (
    -name=> "text4" ,
    -text=> "",
    -left=>350,
    -top=> 195,
    -number=> 1,
    -width=>50,
    -height=>25,
    -tip => "Hello",
   );
$label6= $main->AddLabel
(
-name=> "label6" ,
-text=> "E Value" ,
-left=>280,
```

```
-top=> 250,
-font=>$font,
-height=>40,
);

$text2=$main-> AddCombobox(
        -name =>  'Markets2',
        -pos  =>  [350,250],
        -size =>  [100,100],
        -vscroll =>  1,
        -tabstop =>  1,
        -dropdown =>  1,
);

$main-> Markets2->Add(-36);
 $main-> Markets2->Add(-38);
$button3=$main->AddButton
(
  -name=> "run",
  -text=> "Analyzing",
 -left=>250,
  -top=>330,
  -font=>$font,
    );
   ChildSize($main);
 Win32::GUI::ChangeIcon($main,$icon);

}
#############################################################
#button event to select the
#assmbly file,thats may be single or multie file
sub open1_Click {

  $text=Win32::GUI::GetOpenFileName
      (
              -filemustexist => 1,
              -includefiles => 1,
              -filter=>['All Files - *','*.*','Fasta','*.fasta','TXT',
'*.txt', 'Data', '*.data'],
              -pathmustexist =>0,
              -directory =>'C:\\Project\\',
                  );

$text1->Text($text);
 }
#############################################################
#button event to select the
#WSGR file,must be single file
sub open2_Click{
$List1->Reset();
  @text=Win32::GUI::GetOpenFileName
      (
  -directory =>'C:\\Project\\MainProject\\chr',
                 -filemustexist => 1,
               -defaultextension => "Fasta",
               -includefiles => 1,
              -multisel => 1,
```

```
                -filter =>
               [ 'Fasta', '*.fasta','TXT', '*.txt', 'Data', '*.data',
                 'file', '*.file','All Files - *', '*' ],
             -pathmustexist =>0,
               );

for($i=0;$i<=$#text;++$i)
 {$List1->AddString("$text[$i]");}
 }
#################################################################
# This function manage child window resize.
sub ChildSize {
     my $self = shift;
     my ($width, $height) = ($self->GetClientRect())[2..3];
     # TextField take all client aera
     $self->{Edit}->Resize($width, $height) if exists $self->{Edit};
}
#################################################################
#algrthem taht get the localion of duplication
sub dup_detecte(){
$d=0;
$k=0;
$d_accuracy=5;
$start=0;
@duppx=();
@duppsem=();
loop:for($i=0;$i<=$len;$i=$i+$inc)
      {    $c=0;
        for($j=$start;$j<$#data;$j=$j+1)
           {
           if($i>=$data[$j][0] and $i<=$data[$j][1])
              {
                $dup[$d][0]=$i;
                $dup[$d][1]=$data[$j][0];
                $dup[$d][2]=$data[$j][1];
                $d=$d+1;
               $c++;
              }
            else
              {
             if($i<$data[$j][0])
               {
                 $start=$j;#start of the next

                  $dupx[$k][1]=$c;
                  $dupx[$k][0]=$i;
                  $k++;
                 next loop;#end
               }
            }
           }
      }
$g=0;
 for($j=0;$j<=$#dupx;$j=$j+1)
 {

   if($dupx[$j][1]>$cov-2)
```

```
    {
    $dupmap[$g][0]=$dupx[$j][0];
    $dupmap[$g][1]=$dupx[$j][1];
    $g++;
     }

  }


#************************************ 545

         $s=0;
for($i=0;$i<=$#dupmap;$i++)
  {       $st=$dupmap[$i][0];
          $e=0;
          $end=$dupmap[$i][0];
 for($j=$i+1;$j<=$#dupmap;$j++)
    {
      if($dupmap[$j][0]==$dupmap[$i][0]+(($j-$i)*$inc))
       {
         $e=$j-$i;
         $end=$dupmap[$j][0];
       }
    }
$i=$i+$e;
if($st-1000<0)
{
$st=0;
}
else
 {
 $st=$st-1000;
 }
 if($end+1000>$len)
 {
 $end=$len;
 }
 else
 {
 $end=$end+1000;
 }
$dupp[$s][0]=$st;
$dupp[$s][1]=$end;
$s++;
}



$k=0;
$d=0;
$c=0;
$f=0;

for($p=0;$p<=$#dupp;$p++)
 {
  lop:for($y=$dupp[$p][0];$y<=$dupp[$p][1];$y=$y+$d_accuracy)
    {
```

```
    $c=0;
      for($j=0;$j<$#data;$j++)
          {
          if($y>=$data[$j][0] and $y<=$data[$j][1])
             {

              $duppsem[$f][0]=$data[$j][0];
              $duppsem[$f][1]=$data[$j][1];
              $duppsem[$f][2]=$data[$j][2];
              $duppsem[$f][3]=$data[$j][3];
              $duppsem[$f][4]=$y;

              $f++;
              $c++;
              }else
              {
            if($y<$data[$j][0])
              {
                #$start=$j;start of the next

                 $dupx1[$k][1]=$c;
                 $dupx1[$k][0]=$y;
                 $k++;
                next lop;#end
               }
             }
          }
   }
 }


#***fial filter *********************************************
$g=0;


 for($j=0;$j<=$#dupx1;$j=$j+1)
 {

   if($dupx1[$j][1]>$cov-2)
    {
    $dupmapx[$g][0]=$dupx1[$j][0];
    $dupmapx[$g][1]=$dupx1[$j][1];
    $g++;
     }
 }

#************************************ 545
$s=0;
for($i=0;$i<=$#dupmapx;$i++)
   {       $st=$dupmapx[$i][0];
           $e=0;
           $end=$dupmapx[$i][0];
  for($j=$i+1;$j<=$#dupmapx-1;$j++)
     {
       if($dupmapx[$j][0]==$dupmapx[$i][0]+(($j-$i)*5))
        {
          $e=$j-$i;
```

```
            $end=$dupmapx[$j][0];
          }
       }
          $i=$i+$e;
          $duppx[$s][0]=$st;
          $duppx[$s][1]=$end;
               $s++;
  }

 open("o1",">>draw.txt");

   for($p=0;$p<=$#duppx;$p++)
       {  if($duppx[$p][1]-$duppx[$p][0]>1000)
            {
       print o1 "$duppx[$p][0] $duppx[$p][1]\n";
       }
   }
   close("o1");
      #***********

$sr=0;
$u=0;
for($j=0;$j<=$#duppx;$j++)
{
  if($duppx[$j][1]-$duppx[$j][0]>1000)
  {
         open("out3",">segmain/d$o.txt");
         $o++;
  lo0:for($y=$duppx[$j][0];$y<=$duppx[$j][1];$y=$y+100)
   {
        for($u=$sr;$u<=$#duppsem;++$u)
        {
        if($duppsem[$u][4]==$y){
        print out3 "$duppsem[$u][2]\n";
        }
        else
         {
         if($duppsem[$u][4]>$y)
         {
         $sr=$u;
         next lo0;
           }
         }
        }
   }
 }
}

 }
#############################################################
#read megablast out put file
#and put it in array of datata
sub get_data(){

  open("data","out2.txt");
   $i=0;
 while(<data>)
```

```
  {

($query,$subject,$identity,$len,$mismatches,$gap,$start,$end,$sstart,$s
end,$e,$b)=split(" ",$_);
   $data[$i][0]=$start;
   $data[$i][1]=$end;
   $data[$i][2]= $subject;
   $data[$i][3]= $len;
   $data[$i][4]= $identity;
   $i++;
  }
  close("data");
 }
#################################################################
#get length of assmbly file
#
sub format(){
 $m= $Window->{Client}->AddMDIChild (
        -name => "m" ,
        -text => "format program" ,
        -sizable => 0,
        -controlparent =>1,
        -maximizebox =>1,
        -onTerminate  => sub { return 1;},
        -addexstyle =>WS_EX_STATICEDGE,
        -top => 10,
        -left=> 20,
        -width => 640,
        -height =>200,
                          );
$laddl = $m->AddLabel
  (
  -name=> "la1" ,
  -text=> "Format Data Program " ,
  -left=>300,
  -top=> 15,
  -font=>Win32::GUI::Font->new(
               -name => "Lucida Console",
               -size => 15,
               -italic => 1
               -bold =>1,),);

$he1=$m->AddButton
(
  -text=> "?",
  -left=>258,
  -top=>52,
  -width=>20,
  -onClick=> sub {  my $help = new Win32::GUI::Window(
    -parent =>$m,
    -name   => "W",
    -text   => "help",
    -left     => 250,
    -top      => 140,
    -width  => 200,
    -height => 150,
    -addstyle    =>  WS_BORDER | WS_CAPTION | WS_SYSMENU,
```

```
    -onTerminate  => sub {return 1;},
);
#####################################################
#label that comtain help text
              $lab =$help->AddLabel
             (

             -text=>" Shout Gun Read database should be".
             " a single file containing DNA sequence in fasta".
             " formats that represent the whole genome ".
             " shut gun reads ",
             -left=>10,
             -top=> 10,
              -wrap     => 1,
             -font=>new Win32::GUI::Font(
                         -italic => 1 ,
                         -underline => 1 ,
                         -strikeout => 1,),
            -width=>200,
            -height=>150,
                          );

$help->Show();
},
  -height=>20,
 );

 $lab = $m->AddLabel
(
-name=> "lab" ,
-text=> "Shout Gun Reads Database" ,
-left=>60,
-top=> 55,
-font=>Win32::GUI::Font->new(
               -name => "Lucida Console",
               -size => 10,
               -italic => 1
               -bold =>1,),

           );
################################################
$tx = $m->AddTextfield
   (
    -name=> "tx" ,
    -text=> "",
    -left=>280,
    -top=> 50,
    -width=>230,
    -height=>25,
   );
$but=$m->AddButton
(
  -name=> "open",
  -text=> "Browse",
  -left=>515,
  -top=>52,
  -width=>100,
```

```
   -height=>20,
  );
 $but3=$m->AddButton
 (
  -name=> "run1",
  -text=> "Format",
 -left=>300,
  -top=>100,
  -font=>Win32::GUI::Font->new(
                -name => "Lucida Console",
                -size => 10,
                -italic => 1
                -bold =>1,),
    );
}
sub run1_Click{
$datafile=$tx->Text;
 system( "formatdb -i  $datafile  -p F  " );
}
sub open_Click {

  $fordata=Win32::GUI::GetOpenFileName
     (
            -filemustexist => 1,
            -includefiles => 1,
            -filter=>['All Files - *','*.*','Fasta','*.fasta','TXT',
'*.txt', 'Data', '*.data'],
            -pathmustexist =>0,
            -directory =>'C:\\Project\\',
                );

$tx->Text($fordata);
 }

sub get_length(){
       $len=0;
        open("open",$file);
   while(<open>)
   {
     if($i==0){
     $i=1;}
  else
   {
  chop;
  $len=$len+length($_);
   }
  }
close("open");
}
#############################################################
#sort arry  of data, that fast the process
sub sort_arry(){

for(my $i=0;$i<=$#data;$i++)
{
 for(my $j=$i+1;$j<=$#data;$j++)
 {
```

```
   if($data[$i][0]>$data[$j][0])
   {
   $temp=$data[$i][0];
   $data[$i][0]=$data[$j][0];
   $data[$j][0]=$temp;

   $temp=$data[$i][1];
   $data[$i][1]=$data[$j][1];
   $data[$j][1]=$temp;

   $temp=$data[$i][2];
   $data[$i][2]=$data[$j][2];
   $data[$j][2]=$temp;

   $temp=$data[$i][3];
   $data[$i][3]=$data[$j][3];
   $data[$j][3]=$temp;

   $temp=$data[$i][4];
   $data[$i][4]=$data[$j][4];
   $data[$j][4]=$temp;
   }

  }
 }
 open("dat",">sort.txt");
 for($i=0;$i<$#data;$i++)
 {
 print dat "$data[$i][0] $data[$i][1] $data[$i][2]\n"
 }
close("dat");
}
#############################################################
#get select WSGR and assmbly file and rum megablast to commpary and
#get the significent simaralty genomic relation
sub run_megablast(){
$t=0;
# system( "formatdb -i $location  -p F -o T " );
 print "blast run\n";
$t=system( "MegaBlast -d $location -i $file -o out2.txt -m 8   -v
1000000 -b 1000000 ");
 if($t>0)
 {
 Win32::GUI::MessageBox($W,"Erro in external program
megablast","warning",64,);
 }
 if($t>0)
  {
   Win32::GUI::MessageBox($W,"Error in external program
megablast","warning",64,);

  }

 }
#############################################################
#get select WSGR and assmbly file and rum megablast to commpary and
#get the significent simaralty genomic relation
```

```
sub Draw_dup {

open("data","draw.txt");
 my $i=0;
while(<data>)
 {
 ($start,$end)=split(" ",$_);
  $dataD[$i][0]=$start;
  $dataD[$i][1]=$end;
  $i++;
 }
 close("data");

   $DC = $Win->GetDC;
    $P;
    $B;
    $m=($len)/26/100;
    $P = new Win32::GUI::Pen(
          -color => [10,10, ],
          -width =>2,
      );
       $DC->SelectObject($P);

      for($i=0;$i<=$#dataD;++$i)
      {

       $Win->AddButton
                   (
                     -name=>'dd'.$i,
                     -text=>'Dup',
                     -left=>(($dataD[$i][1]-$dataD[$i][0])/10+20),
                     -top=>($i+1)*30-8,


                    );
         $Win->AddButton
                   (
                     -name=>'ddd'.$i,
                     -text=>'Seg',
                     -left=>(($dataD[$i][1]-$dataD[$i][0])/10+60),
                     -top=>($i+1)*30-8,  );
       $Win->AddLabel
                   (
                     -text=>$dataD[$i][1],
                     -left=>(($dataD[$i][1]-$dataD[$i][0])/10)-10,
                     -top=> ($i+1)*30-15,
                        ) ;

      $Win->AddLabel
                   (
                     -text=> $dataD[$i][0],
                     -left=>100,
                     -top=> ($i+1)*30-15,
                    );

     $DC->Rectangle(100,($i+1)*30,($dataD[$i][1]-
$dataD[$i][0])/10,($i+1)*30+10);
```

```
        }
    }
    sub Window_Terminate {
    Win32::GUI::Show($DOS);

    exit(0);
        return -1;
    }
    sub Splash_Terminate {
        return 1;
    }
    sub progras{
     $main->Disable();
      $probarform1  = new Win32::GUI::Window
            ( -owner=> $main,
              -name => "probarform1" ,
              -text =>" Process Analyzing" ,
              -sizable => 0,
              -top => 200,
              -left=>100,
              -parent =>$main,
              -width => 500,
              -height =>150,
              -maximizebox => 0,
              -sysmenu => 0
             );
           $probar1 = $probarform1->AddProgressBar
               (
                -name=>"probar1",
                -left =>40,
                -top =>50,
                -width =>400,
                -height => 30,

                );


            $label46 = $probarform1->AddLabel
                       (
                       -name=> "label1" ,
                       -text=>"Please Wait...",
                       -left=>50,
                       -top=>20,
                       -width=>100,
                       -height=>20,
                       );
            $probar1->SetStep(20);
            $probar1->SetBkColor([255,255,255]);
            $probar1->SetPos(2);
            $probarform1->Show;

    }
    sub run_Click{
    $cov=$text3->Text;

     if(($text3->Text() eq  "")|($text1->Text() eq "") | ($text4->Text() eq
```

```
"") | ($cov le 1) | $#text<0)
                    {
                        Win32::GUI::MessageBox($W,"you maust enter
all filed","warning",64,);
                    }
       else
        {
        $cov=$cov*2;
         $location=$text1->Text;
         if (open("dfd",$location)){

           open("dr",">draw.txt"); close("dr");
                    @data;
                     $len;
                     $inc=$text4->Text;
                     $evalue;
           if($#text==0)
                {
                 $df=$#text+1;
                 Win32::GUI::MessageBox($W,"the programm Analyze $df
file you must be patient that take Minutes","Analyze file",64,);
                &progras();
                 $probar1->StepIt();
                 $file=$text[0];
                 &run_megablast($location,$file);
                if($t>0)
                  {  $main->Enable();
                     $probarform1->Hide();
                     return ;
                  }
                 $probar1->StepIt();
                 &get_data(@data);
                 $probar1->StepIt();
                 &get_length($len,$file);
                 $probar1->StepIt();
                 &sort_arry(@data);
                 &dup_detecte(@data,$len);
                 $probar1->StepIt();
                 $main->Enable();
                 $probarform1->Hide();
                 &draw();
                 &Draw_dup();
                 }
              else
                {
                 $df=$#text;
                 Win32::GUI::MessageBox($W,"the programm Analyze $df
file you must be patient that take Minutes","Analyze file",64,);
                  &progras();
                for(my $i=1;$i<=$#text;++$i){
                                $probar1->StepIt();
                                  print "finash $i\n";
                                 $file=$text[0].'\\'.$text[$i];
                                &run_megablast($location,$file);
                                 if($t>0)
                                                {  $main->Enable();
                                                   $probarform1-
```

```
>Hide();

                                                    return ;
                                                }
                                    $probar1->StepIt();
                                    &get_data(@data);
                                    &get_length($len,$file);
                                    $probar1->StepIt();
                                    &sort_arry(@data);
                                    &dup_detecte(@data,$len);
                                    $probar1->StepIt();
                                }
                    $probar1->StepIt();
                    $main->Enable();

                    $probarform1->Hide();
                    &draw();
                    &Draw_dup();
                    }#########


    }else{ Win32::GUI::MessageBox($W,"$location File not found, please
verify the correct file name was given","Erro",64,);}
  }#else
 }#end of sub
sub  dd0_Click{
$x=1;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$a1[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

     if($i !=$x)
      {
       open("file","\segmain/d$i.txt") or return 1;
        while(<file>)
         {
           chop;
          for($j=0;$j<$#a1;$j++)
          {
            if($_ eq $a1[$j])
            {
            $dat[$gy]=$i;
            $gy++;
            }

         }
        }
     }
```

```
    }
    $gf=0;
  for($ni=0;$ni<$#dat;$ni++)
  {$c=0;
   $el=0;
   for($in=$ni+1;$in<$#dat;$in++)
       {
          if($dat[$ni]==$dat[$in])
          {
          $el=$in-$ni;
          $c++;
          }

       }
   $ni=$ni+$el;
   $ddat[$gf][0]=$dat[$ni];
   $ddat[$gf][1]=$c++;
   $gf++;
  }
     $max=$ddat[0][1];
     $po=$ddat[0][0];
  for($ni=0;$ni<=$#ddat;$ni++)
  {  if($max<$ddat[$ni][1])
       {
       $max=$ddat[$ni][1];
       $po=$ddat[$ni][0];
       }
  }

 Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub

 sub  dd1_Click{
$x=2;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$a1[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

     if($i !=$x)
      {
       open("file","\segmain/d$i.txt") or return 1;
        while(<file>)
         {
           chop;
         for($j=0;$j<$#a1;$j++)
         {
```

```
            if($_ eq $al[$j])
            {
            $dat[$gy]=$i;
            $gy++;
            }

          }
         }
       }
    }
    $gf=0;
  for($ni=0;$ni<$#dat;$ni++)
  {$c=0;
   $el=0;
   for($in=$ni+1;$in<$#dat;$in++)
       {
          if($dat[$ni]==$dat[$in])
          {
          $el=$in-$ni;
          $c++;
          }

       }
   $ni=$ni+$el;
   $ddat[$gf][0]=$dat[$ni];
   $ddat[$gf][1]=$c++;
   $gf++;
  }
     $max=$ddat[0][1];
     $po=$ddat[0][0];
  for($ni=0;$ni<=$#ddat;$ni++)
  {  if($max<$ddat[$ni][1])
       {
       $max=$ddat[$ni][1];
       $po=$ddat[$ni][0];
       }
  }

 Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub

 sub  dd2_Click{
$x=3;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$al[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {
```

```
      if($i !=$x)
       {
        open("file","\segmain/d$i.txt") or return 1;
         while(<file>)
          {
            chop;
           for($j=0;$j<$#a1;$j++)
           {
             if($_ eq $a1[$j])
             {
             $dat[$gy]=$i;
             $gy++;
             }

           }
          }
       }
    }
    $gf=0;
 for($ni=0;$ni<$#dat;$ni++)
 {$c=0;
  $e1=0;
  for($in=$ni+1;$in<$#dat;$in++)
      {
         if($dat[$ni]==$dat[$in])
         {
         $e1=$in-$ni;
         $c++;
         }

      }
  $ni=$ni+$e1;
  $ddat[$gf][0]=$dat[$ni];
  $ddat[$gf][1]=$c++;
  $gf++;
 }
    $max=$ddat[0][1];
    $po=$ddat[0][0];
for($ni=0;$ni<=$#ddat;$ni++)
{  if($max<$ddat[$ni][1])
       {
       $max=$ddat[$ni][1];
       $po=$ddat[$ni][0];
       }
}

 Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub

 sub  dd3_Click{
$x=4;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
```

```
{
chop;
$al[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

     if($i !=$x)
      {
       open("file","\segmain/d$i.txt") or return 1;
        while(<file>)
         {
           chop;
          for($j=0;$j<$#al;$j++)
          {
            if($_ eq $al[$j])
            {
            $dat[$gy]=$i;
            $gy++;
            }

          }
         }
      }
  }
  $gf=0;
for($ni=0;$ni<$#dat;$ni++)
{$c=0;
 $el=0;
 for($in=$ni+1;$in<$#dat;$in++)
     {
        if($dat[$ni]==$dat[$in])
        {
        $el=$in-$ni;
        $c++;
        }

     }
 $ni=$ni+$el;
 $ddat[$gf][0]=$dat[$ni];
 $ddat[$gf][1]=$c++;
 $gf++;
}
   $max=$ddat[0][1];
   $po=$ddat[0][0];
for($ni=0;$ni<=$#ddat;$ni++)
{  if($max<$ddat[$ni][1])
      {
      $max=$ddat[$ni][1];
      $po=$ddat[$ni][0];
      }
}

Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
```

```
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub

 sub  dd4_Click{
$x=5;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$a1[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

     if($i !=$x)
      {
       open("file","\segmain/d$i.txt") or return 1;
        while(<file>)
         {
           chop;
          for($j=0;$j<$#a1;$j++)
          {
            if($_ eq $a1[$j])
            {
            $dat[$gy]=$i;
            $gy++;
            }

          }
         }
     }
  }
  $gf=0;
for($ni=0;$ni<$#dat;$ni++)
{$c=0;
 $e1=0;
 for($in=$ni+1;$in<$#dat;$in++)
     {
        if($dat[$ni]==$dat[$in])
        {
        $e1=$in-$ni;
        $c++;
        }

     }
 $ni=$ni+$e1;
 $ddat[$gf][0]=$dat[$ni];
 $ddat[$gf][1]=$c++;
 $gf++;
}
   $max=$ddat[0][1];
   $po=$ddat[0][0];
```

```
 for($ni=0;$ni<=$#ddat;$ni++)
 {  if($max<$ddat[$ni][1])
       {
       $max=$ddat[$ni][1];
       $po=$ddat[$ni][0];
       }
 }

 Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub

 sub  dd5_Click{
$x=6;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$a1[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

    if($i !=$x)
     {
      open("file","\segmain/d$i.txt") or return 1;
       while(<file>)
        {
          chop;
         for($j=0;$j<$#a1;$j++)
         {
           if($_ eq $a1[$j])
           {
           $dat[$gy]=$i;
           $gy++;
           }

         }
        }
    }
  }
  $gf=0;
for($ni=0;$ni<$#dat;$ni++)
{$c=0;
 $el=0;
 for($in=$ni+1;$in<$#dat;$in++)
    {
       if($dat[$ni]==$dat[$in])
       {
       $el=$in-$ni;
       $c++;
       }
```

```
        }
    $ni=$ni+$el;
    $ddat[$gf][0]=$dat[$ni];
    $ddat[$gf][1]=$c++;
    $gf++;
   }
      $max=$ddat[0][1];
      $po=$ddat[0][0];
  for($ni=0;$ni<=$#ddat;$ni++)
  {  if($max<$ddat[$ni][1])
        {
        $max=$ddat[$ni][1];
        $po=$ddat[$ni][0];
        }
  }

 Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub
 sub  dd6_Click{
$x=7;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$al[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

     if($i !=$x)
      {
       open("file","\segmain/d$i.txt") or return 1;
        while(<file>)
         {
           chop;
          for($j=0;$j<$#al;$j++)
          {
            if($_ eq $al[$j])
            {
            $dat[$gy]=$i;
            $gy++;
            }

          }
         }
     }
  }
  $gf=0;
for($ni=0;$ni<$#dat;$ni++)
{$c=0;
```

```
    $el=0;
    for($in=$ni+1;$in<$#dat;$in++)
        {
           if($dat[$ni]==$dat[$in])
           {
           $el=$in-$ni;
           $c++;
           }

        }
    $ni=$ni+$el;
    $ddat[$gf][0]=$dat[$ni];
    $ddat[$gf][1]=$c++;
    $gf++;
   }
      $max=$ddat[0][1];
      $po=$ddat[0][0];
  for($ni=0;$ni<=$#ddat;$ni++)
  {  if($max<$ddat[$ni][1])
        {
        $max=$ddat[$ni][1];
        $po=$ddat[$ni][0];
        }
  }

  Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub

 sub  dd7_Click{
$x=8;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$a1[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

     if($i !=$x)
      {
       open("file","\segmain/d$i.txt") or return 1;
        while(<file>)
         {
           chop;
          for($j=0;$j<$#a1;$j++)
          {
            if($_ eq $a1[$j])
            {
            $dat[$gy]=$i;
            $gy++;
```

```
            }

          }
         }
       }
    }
    $gf=0;
  for($ni=0;$ni<$#dat;$ni++)
  {$c=0;
   $e1=0;
   for($in=$ni+1;$in<$#dat;$in++)
       {
          if($dat[$ni]==$dat[$in])
          {
          $e1=$in-$ni;
          $c++;
          }

       }
   $ni=$ni+$e1;
   $ddat[$gf][0]=$dat[$ni];
   $ddat[$gf][1]=$c++;
   $gf++;
  }
     $max=$ddat[0][1];
     $po=$ddat[0][0];
  for($ni=0;$ni<=$#ddat;$ni++)
  {  if($max<$ddat[$ni][1])
       {
       $max=$ddat[$ni][1];
       $po=$ddat[$ni][0];
       }
  }

 Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub

 sub  dd8_Click{
$x=9;
$d=0;
$gy=0;
open("pot","\segmain/d$x.txt") or  return 1;
while(<pot>)
{
chop;
$a1[$d]=$_;
$d++;
}

close("pot");
for($i=1;$i<=$#dataD+1;$i++)
   {

     if($i !=$x)
      {
       open("file","\segmain/d$i.txt") or return 1;
```

```
        while(<file>)
         {
           chop;
          for($j=0;$j<$#a1;$j++)
          {
            if($_ eq $a1[$j])
            {
            $dat[$gy]=$i;
            $gy++;
            }

          }
         }
       }
    }
    $gf=0;
  for($ni=0;$ni<$#dat;$ni++)
  {$c=0;
   $e1=0;
   for($in=$ni+1;$in<$#dat;$in++)
       {
          if($dat[$ni]==$dat[$in])
          {
          $e1=$in-$ni;
          $c++;
          }

       }
   $ni=$ni+$e1;
   $ddat[$gf][0]=$dat[$ni];
   $ddat[$gf][1]=$c++;
   $gf++;
  }
     $max=$ddat[0][1];
     $po=$ddat[0][0];
 for($ni=0;$ni<=$#ddat;$ni++)
 {  if($max<$ddat[$ni][1])
       {
       $max=$ddat[$ni][1];
       $po=$ddat[$ni][0];
       }
 }

 Win32::GUI::MessageBox($W,"$dataD[$x-1][0] $dataD[$x-1][1] --->
$dataD[$po-1][0] $dataD[$po-1][1]","relation",64,);
}#end of sub
 ###open the segmant the contin the duplication postion
sub finaldraw(){
}
 sub  ddd0_Click{
$x1=1;
&fileseg($x);

}
 sub  ddd1_Click{
$x1=2;
&fileseg($x);
```

```
}
 sub  ddd2_Click{
$x1=3;
&fileseg($x);
}
 sub  ddd3_Click{
$x1=4;
&fileseg($x);

}
 sub  ddd4_Click{
$x1=5;
&fileseg($x);
}
 sub  ddd5_Click{
$x1=6;
&fileseg($x);
}
sub  ddd6_Click{
$x1=7;
&fileseg($x);

}
 sub  ddd7_Click{
$x1=8;
&fileseg($x);
}
 sub  ddd8_Click{
$x1=9;
&fileseg($x);
}

sub fileseg(){
$c=0;
open("fx","\segmain/d$x1.txt") or return 1;
while(<fx>){
chop;
$af[$c]=$_;
$c++;
}
close("fx");

$ma=new Win32::GUI::Window (
        -name => "ma" ,
        -parent =>$Win,
        -text => "segments" ,
        -sizable => 0,
        -controlparent =>1,
        -maximizebox =>0,
        -onTerminate  => sub { return 1;},
        -addexstyle =>WS_EX_STATICEDGE,
        -top => 200,
        -left=> 300,
        -width => 200,
        -height =>300,
       );
```

```
  $List = $ma->AddListbox(
      -name => "List",
      -left => 0,
      -top  => 0,
      -height =>288,
      -menu => 1,
      -tabstop => 1,
      -group => 1,
      -width => 195, -multisel => 0,
      -addstyle => WS_VSCROLL | WS_VISIBLE | WS_CHILD,
      -foreground => [255, 255, 255],
      -background => [64, 64, 64],
  );

  $List->Reset();
  for($i=0;$i<=$#af;++$i)
   {
   $List->AddString("$af[$i]");
   }
  $ma->Show();

  }

  sub about{
  use Win32::GUI;
  my ($width,$height);
  my $mainwin;

  #try to load the splash bitmap from the exe that is running
  my $splashimage= new Win32::GUI::Bitmap('SPLASH');
  unless ($splashimage) {
    #bitmap is not in exe, load from file
    $splashimage= new Win32::GUI::Bitmap('SPLASH.bmp');
    die 'could not find splash bitmap' unless $splashimage;
    #get the dimensions of the bitmap
    ($width,$height)        = $splashimage->Info();
    }

  #create the splash window
  my $splash      = new Win32::GUI::Window (
     -name        => "Splash",
     -text        => "Duplication Finder",
     -height      => $height,
     -width       => $width,
     -left        => 100,
     -parent=>$main,

     -top         => 100,

     -popstyle    => WS_CAPTION | WS_THICKFRAME,
     -addexstyle => WS_EX_TOPMOST
  );


  my $bitmap     = $splash->AddLabel(
      -name      => "Bitmap",
      -left      => 0,
```

```
    -top        => 0,
    -width      => $width,
    -height     => $height,
    -bitmap     => $splashimage,

);
$bitmap->SetImage( $splashimage );


#center the splash and show it
$splash->Center;
$splash->Show();
#call do events - not Dialog - this will display the window and let us
#build the rest of the application.

#A good way of building your application is to keep everything in
packages, and eval those
#into scope in this phase. In this case, we'll create the main window
and sleep to simulate
#some work.

$splash->Show;
Win32::GUI::Dialog();
}
```