CHAPTER 3

PROGRAM DEVELOPMENT

3 Program Development

3.1 Introduction

The program development process is a series of activities that are necessary for the successful development of a computer program, however simple or complex. The program development process can be divided into six stages:

- Defining the problem
- Designing the program
- Coding the program
- Testing the program
- Installing and maintaining the program
- Documenting the program

3.2 Defining the Problem

Before a problem can be correctly solved, it must be clearly defined and understood by the programmer. Once the problem is clearly defined and understood the programmer can use the program development tools available to effectively plan a solution.

> *"Problem definition is a common but difficult task because true problems are often disguised in a variety of ways. It takes a skilful individual to analyse a situation and extract the real problem from a sea of information. Ill-defined or poorly posed problems can lead novice (and not so novice) engineers down the wrong path to a series of impossible or spurious solutions. Defining the 'real problem' is critical to finding a workable solution."*
>
> *– H. Scott Fogler and Steven E. LeBlanc*

The problem should be carefully analyzed to extract the following information:

(a) The **Input** required by the program.
(b) The **Output** or end result required .
(c) The **Process** - a list of actions to be performed to achieve the desired output.

3.2.1 IPO Chart

Programmers use **IPO** (**I**nput, **P**rocessing, **O**utput) charts to organize and summarize the results of a problem definition. The output is the first data added to the IPO Chart, you can then ask what inputs the program need to produce the required output. The next set of data to be added to the IPO Chart is the input. Once you have determined the output you need and the input required to produce the output, the processing portion of the chart can be determined. **Processing** is simply the series of actions you must perform on the input to get the required output.

Suppose our problem is to create a cake.

- Our output is a cake
- Our input therefore are the ingredients needed for the cake
- The processing is the series of activities that must be performed on the inputs (ingredients) to create the cake

The above steps clearly define the problem of making a cake.

3.2.2 Input Statement

We will use the word "input" to create input statements for our **IPO** chart in this text book. Other texts books might use the words, GET, SET, ACCEPT, or READ as their input word.

An input statement will use the following rule:

input VAR_NAME

This means to accept an input and store it in the variable VAR_NAME. For example:

input Numb1

This means to accept an input and store it in the variable Numb1.

3.2.3 Output Statement

We will use the word "print" to create an output statement for the IPO chart. The statement will use the following rule:

print "Phrase" or print VAR_NAME

This means to print the phrase enclosed between the quotes or print the content of VAR_NAME. For example:

print "Hello World" or print Numb1

where our phrase is "Hello World" and Numb1 is a variable.

3.2.1 IPO Chart

Programmers use **IPO** (**I**nput, **P**rocessing, **O**utput) charts to organize and summarize the results of a problem definition. The output is the first data added to the IPO Chart, you can then ask what inputs the program need to produce the required output. The next set of data to be added to the IPO Chart is the input. Once you have determined the output you need and the input required to produce the output, the processing portion of the chart can be determined. **Processing** is simply the series of actions you must perform on the input to get the required output.

Suppose our problem is to create a cake.

- Our output is a cake
- Our input therefore are the ingredients needed for the cake
- The processing is the series of activities that must be performed on the inputs (ingredients) to create the cake

The above steps clearly define the problem of making a cake.

3.2.2 Input Statement

We will use the word "input" to create input statements for our **IPO** chart in this text book. Other texts books might use the words, GET, SET, ACCEPT, or READ as their input word.

An input statement will use the following rule:

input VAR_NAME

This means to accept an input and store it in the variable VAR_NAME. For example:

input Numb1

This means to accept an input and store it in the variable Numb1.

3.2.3 Output Statement

We will use the word "print" to create an output statement for the IPO chart. The statement will use the following rule:

print "Phrase" or print VAR_NAME

This means to print the phrase enclosed between the quotes or print the content of VAR_NAME. For example:

print "Hello World" or print Numb1

where our phrase is "Hello World" and Numb1 is a variable.

3.2.4 PROCESSING STATEMENT

For processing, our IPO chart we will use the following convention:

VAR_NAME = Expression

where *expression* is a formula or a value. For example:

Numb1 = 20
Sum = Numb1 + 30

are valid processing statements. *Numb1 = 20*, means store the value 20 in the variable *Numb1*. *Sum = Numb1 + 30* means add 30 to the value in variable *Numb1* and store the result in the variable *Sum*.

PRACTICE EXAMPLE 3

Write a program that calculates the sum and average of two numbers and print the average.

SOLUTION

DISCUSSION

Suppose you were asked to add two numbers mentally. The first thing you would do is ask for the two numbers. When you are given the two numbers, you would add them, then divide the result by (2) to get the average. You would then communicate the result to the person who asked you for it. The **IPO Chart** defines the problem as shown below:

IPO Chart	INPUT	PROCESS	OUTPUT
	Numb1 Numb2	input Numb1 input Numb2 Sum = Numb1+Numb2 Average = Sum/2 print Average	Average

Figure 3.1 - IPO Chart for practice example 3

PRACTICE EXAMPLE 4

Write a program that calculates and prints the tax paid by a worker and net pay received, given the hourly rate and the number of hours worked for the week. Tax is equal to 30% of total pay and net pay is equal to total pay minus tax.

SOLUTION

DISCUSSION

The ouputs required are: **tax** and **net pay**. The inputs are the **hourly rate** and **hours worked for the week**. The tax rate (30%) is not an input. We will make it a constant because it is the same for all calculations.

To determine the process, we need to ask the question; What must I do to the input and the data that I know (constants) to get the required output. We need to do the folllowing:

1. Compute the TotalPay (HourlyRate*HoursWorked)
2. Compute the tax paid (TaxRate*TotalPay)
3. Compute the net pay (TotalPay - Tax)

IPO Chart	Input	Process	Output
	HourlyRate HoursWorked	input HourlyRate input HoursWorked TotalPay = HourlyRate * HoursWorked Tax = TaxRate * TotalPay NetPay = TotalPay – Tax print Tax print NetPay	Tax NetPay

Figure 3.2 - IPO Chart for practice example 4

3.3 Designing the Program

Once we have clearly defined the problem, several programming tools are now available to assist the programmer in designing a solution for the problem. One should note that for a given problem, several solutions may exist. The programmer must examine the alternative solutions and choose the best one.

Some of the tools available to the programmer during the design phase are:

- Algorithms
- Flowcharts
- Pseudocodes
- Decision tables
- Structure charts
- HIPO charts

The programmer can use some or all of the above tools in the design phase depending on the complexity of the problem to be solved. For the purpose of this text our main design tool will be Pseudocode along with a basic introduction to Flowchart.

3.3.1 PSEUDOCODE

After the creation of the IPO Chart, the next step in creating a solution is to develop a set of instructions for the computer, called a pseudocode. A Pseudocode is similar to an algorithm and sometime the words are used interchangebly. A pseudocde is a informal programming language that is very similar to the High Level programming language but without the strictness of the syntax of a High Level Language. We use the term "informal" because the rules and phrases used for pseudocode statements are subjective and depend on the style and preference of the programmer. The input, output and processing statements for pseudocode will follow the conventions used by the IPO chart in the previous section.

The computer program is the translation of the pseudocode to the High Level programming language. The pseudocode is created using the information in the process column of the IPO Chart as a guide.

PRACTICE EXAMPLE 5

Write a program that calculates the sum and average of two numbers and prints the average.

SOLUTION

Based on the IPO chart (Figure 3.1) our pseudocode becomes:

Pseudocode

```
Step 1.   input Numb1
Step 2.   input Numb2
Step 3.   Sum = Numb1+Numb2
Step 4.   Average = Sum/2
Step 5.   print Average
```

Figure 3.3 - Pseudocde for practice example 5

PRACTICE EXAMPLE #6

Write a program that calculates and prints the tax paid by a worker and net pay received, given the hourly rate and the number of hours worked for the week. Tax is equal to 30% of total pay and net pay is equal to total pay minus tax.

SOLUTION

Based on the IPO chart (Figure 3.2), our pseudocode becomes:

Pseudocode

```
Step 1.   input HourlyRate
Step 2.   input HoursWorked
Step 3.   TotalPay = HourlyRate * HoursWorked
Step 4.   Tax = TaxRate * TotalPay
Step 5    NetPay = TotalPay – Tax
Step 6.   print Tax
Step 7.   print NetPay
```

Figure 3.4 - Pseudocde for practice example 6

3.3.2 FLOWCHART

A Flowchart is a graphic form of a pseudocode or algorithm. Programming instructions and control structures are represented by symbols instead of alphanumeric phrases. The following table outlines the flowchart symbols we will use in this text.

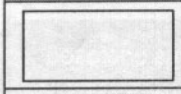
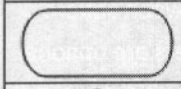
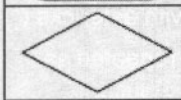
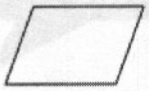
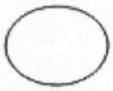
Flowchart Symbol	Name	Description
	Process	An operation or action
	Terminator	A start or stop point in a program
	Decision	A question or branch in a program
	Input/output	Indicates data inputs or outputs to or from a program
	Flow Line	Indicates direction of program flow
	Connector	A jump from one point to another

Figure 3.5 - Table of Flowchart Symbol

Some rules for drawing flowcharts:

1. You should write the instruction inside the blocks
2. I there is something you need to remember you can write a note beside a block. Test values of variables also can be placed beside flowchart blocks.
3. A flowchart always starts at the top of the page and flows to the bottom. If you need more than one page. On and off-page connectors are used to connect parts of the flowchart. A flowchart should not flow up, or side ways, or all over the page
4. Use a template and a straightedge, to draw the flow chart. When you use a template, the symbols are the same size and your flowchart will be neater and easier to read.
5. Make the blocks big enough to write instructions, so they can be easily read.
7. Have plenty of paper on hand since the final copy of the flowchart normally will not be the first draft.
8. Use a pencil with a large eraser.

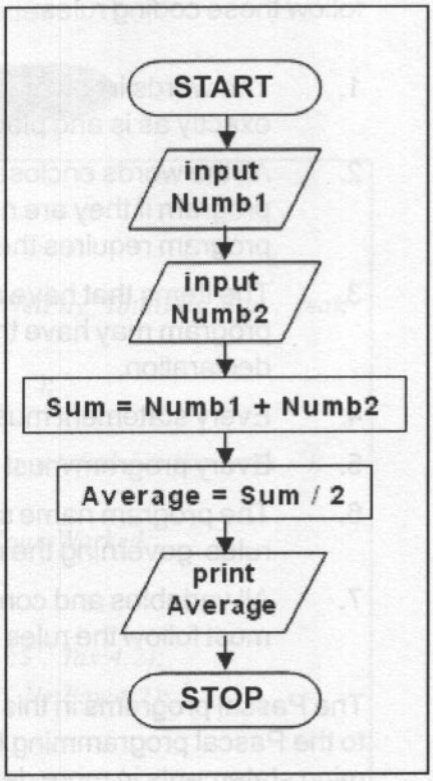


Figure 3.6 - Flowchart design for practice example 3

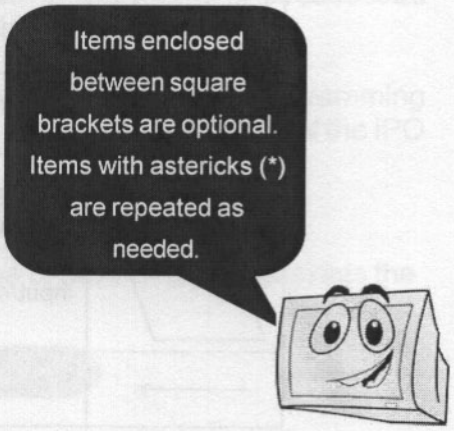
3.4 Coding the Solution in Pascal

The next step is to translate the pseudocode to the Pascal Programming Language.

3.4.1 Structure Of A Pascal Program

Every Pascal Program has the following basic outline:

```
program PROGRAMNAME (input, output);
[const]
        [CONSTANTNAME = CONSTANTVALUE;]*
[var]
        [VARNAME : DATA TYPE;]*
begin
        [PROGRAM STATEMENT;]*
end.
```



3.4.2 Coding Rule For Pascal

Refering to the outline in section 3.4.1. when converting a pseudocode to Pascal codes, follow these coding rules:

1. The words in lower case letters are compulsory in every program and must be spelt exactly as is and placed in the location indicated.
2. All the words enclosed in square brackets are optional and only inserted in the program if they are needed. For example, the word "***const***" is only inserted if the program requires the use of a constant.
3. The items that have an asterisk (*) are repeated as needed. For example, a program may have three statements, two variable declarations and one constant declaration.
4. Every statement must end with a semi colon (;).
5. Every program must end with a full stop (.) placed after the end.
6. The program name supplied by the programmer is compulsory and must follow the rules governing the naming of identifiers as outlined on page 8.
7. All variables and constants used in the program must be declared and their names must follow the rules governing naming of identifiers.

The Pascal programs in this chapter, serve as a illustrations and also as a basic introduction to the Pascal programming language. In the next chapter, we will examine Pascal programming statements in more detail.

PRACTICE EXAMPLE 7

Convert the pseudocode in Figure 3.3 to Pascal codes.

SOLUTION

Pascal Codes

Figure 3.7 - Pascal code for practice example 7

```
program Average (input, output);
var
        Numb1, Numb2, Sum, Avg: real;
begin
        write('Enter first number  :');
        readln (Numb1);
        write('Enter second number :');
        readln (numb2);
        Sum := Numb1 + Numb2;
        Avg := Sum/2;
        Writeln ('Average of the two numbers is  =:',Avg:3:2);
        readln;
end.
```

PRACTICE EXAMPLE 8

Convert the pseudocode in Figure 3.2 to Pascal codes.

SOLUTION

Pascal Codes

Figure 3.8 - Pascal code for practice example 8

```
program Salary(input, output);

var
        HourlyRate, HoursWorked, NetPay, TotalPay, Tax : real;
begin
        write('Enter the hourly rate  :');
        readln (HourlyRate);
        write('Enter number of hours worked for the week :');
        readln (HoursWorked);
        TotalPay := HourlyRate*HoursWorked;
        Tax := 0.3*TotalPay;
        NetPay := TotalPay – Tax;
        Writeln ('The tax amount is :$', Tax:4:2);
        Writeln ('The Net Pay is :$', NetPay:4:2);
        readln;
end.
```

PRACTICE EXAMPLE 9

Write a program that reads a eight bit binary number one bit at a time and converts it to its decimal value. For your solution, include: IPO Chart, Pseudocode and Pascal Codes.

SOLUTION

DISCUSSION

The output required is a **decimal number**. The input is the **eight bits**.

To determine the process, we assume that the bits of the binary number are entered from left to right. For example, consider the binary number: 00100100. The first bit entered would be 0.

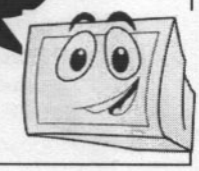
1. We need to multiply each bit by its position number to get its base value
2. We need to add all the base values to get the decimal value.

The position number for each bit from left to right are: 128, 64, 32, 16, 8, 4, 2, 1.

IPO Chart

Figure 3.9 - IPO Chart for practice example 9

Input	Process	Output
Bit1	BaseValue1 = Bit1 * PositionValue1	DecimalValue
Bit2	BaseValue2 = Bit2 * PositionValue2	
Bit3	BaseValue3 = Bit3 * PositionValue3	
Bit4	BaseValue4 = Bit4 * PositionValue4	
Bit5	BaseValue5 = Bit5 * PositionValue5	
Bit6	BaseValue6 = Bit6 * PositionValue6	
Bit7	BaseValue7 = Bit7 * PositionValue7	
Bit8	BaseValue8 = Bit8 * PositionValue8	
	DecimalValue = BaseValue1 + BaseValue2 + BaseValue3 + BaseValue4 + BaseValue5 + BaseValue6 + BaseValue7 + BaseValue8	



Pseudocode

Figure 3.10 - Pseudocde for practice example 9

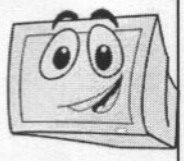
```
BaseValue1 = Bit1 * PositionValue1
BaseValue2 = Bit2 * PositionValue2
BaseValue3 = Bit3 * PositionValue3
BaseValue4 = Bit4 * PositionValue4
BaseValue5 = Bit5 * PositionValue5
BaseValue6 = Bit6 * PositionValue6
BaseValue7 = Bit7 * PositionValue7
BaseValue8 = Bit8 * PositionValue8

DecimalValue = BaseValue1 + BaseValue2 + BaseValue3 + BaseValue4 +
               BaseValue5 + BaseValue6 + BaseValue7 + BaseValue8
```

Pascal Codes

Figure 3.11 - Pascal code for practice example 9

```
program Salary(input, output);
const
        PositionValue1 = 128;
        PositionValue2 = 64;
        PositionValue3 = 32;
        PositionValue4 = 16;
        PositionValue5 = 8;
        PositionValue6 = 4;
        PositionValue7 = 2;
        PositionValue8 = 1;
var
        BaseValue1, BaseValue2, BaseValue3, BaseValue4, BaseValue5, BaseValue6, BaseValue7,
        BaseValue8, Bit1, Bit2, Bit3, Bit4, Bit5, Bit6, Bit7, Bit8, DecimalValue:integer;
begin
        write('Enter first bit 1 or 0  : '); readln(Bit1);
        write('Enter second bit 1 or 0  : '); readln(Bit2);
        write('Enter third bit 1 or 0  : '); readln(Bit3);
        write('Enter fourth bit 1 or 0  : '); readln(Bit4);
        write('Enter fifth bit 1 or 0  : '); readln(Bit5);
        write('Enter sixth bit 1 or 0  : '); readln(Bit6);
        write('Enter seventh bit 1 or 0  : '); readln(Bit7
        write('Enter eight bit 1 or 0  : '); readln(Bit8);

        BaseValue1 := Bit1*PositionValue1;
        BaseValue2 := Bit2*PositionValue2;
        BaseValue3 := Bit3*PositionValue3;
        BaseValue4 := Bit4*PositionValue4;
        BaseValue5 := Bit5*PositionValue5;
        BaseValue6 := Bit6*PositionValue6;
        BaseValue7 := Bit7*PositionValue7;
        BaseValue8 := Bit8*PositionValue8;

        DecimalValue :=     BaseValue1+ BaseValue2+ BaseValue3+
                            BaseValue4+BaseValue5+ BaseValue6+
                            BaseValue7+ BaseValue8;
        writeln('The decimal eguivalent of the binary number is  :',DecimalValue);
        readln;
end.
```

You can place an instruction on more than one line as long as you terminate it with a semi-colon on the last line of the instruction. You can also place more than one instruction on the same line as long as each instruction ends with a semi-colon.



Review Question

1. List the stages in the program development process..

2. List the information the programmer must extract from the problem definition stage of the program development process..

3. Given the following programming problem, create a IPO chart.

   A company pays its workers a rate of US$200.00 per hour. Write a program that calculates a worker's monthly salary after income tax is deducted in Jamaican Dollars. The program accepts as input the numbers of hours worked for the month and the current exchange rate to convert US dollars to Jamaican dollars. The income tax rate is 30% of salary.

4. Give two words other than the ones introduced in this text that are suitable to be used for input and output in a pseudocde.

5. Develop your own convention for a processing statement other than the one used in this chapter.

6. Rewrite the IPO chart in Figure 3.2 using the input, output and processing information you created in Questions, 4 and 5 above.

7. State two major differences between a High Level language and a pseudocode.

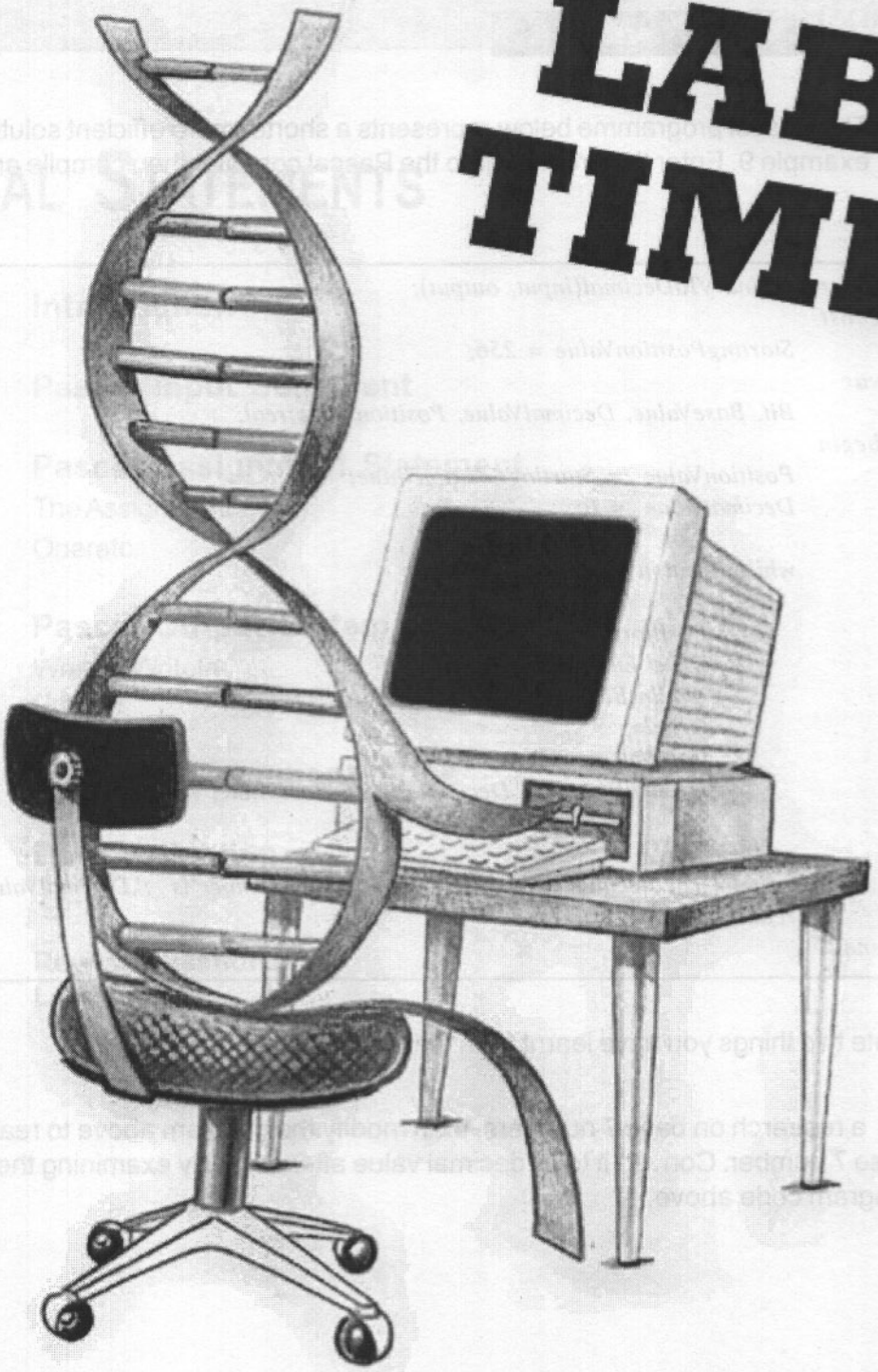
8. Explain how the following programming tools relate to each other: IPO chart, pseudocode, flowchart and Pascal codes.

9. Convert the following pseudocode to Pascal codes:

```
input CostOfItem
input Quantity

Cost = CostOfItem*Quantity* 1.175
Discount = Cost * 0.15

FinalCost = Cost - Discount

print FinalCost
```

10. Create flowcharts for the IPO Chart in Figures 3.2 and 3.9.

LAB TIME



TRY THIS

1. Compile and run the programs in Figure 3.7, Figure 3.8 and Figure 3.11.

LEARNING BY EXAMPLE

2. The Pascal programme below represents a shorter more efficient solution to practice example 9. Enter the program into the Pascal compiler then compile and run it.

```
program BinaryToDecimal(input, output);
const
        StartingPositionValue = 256;
var
        Bit, BaseValue, DecimalValue, PositionValue:real;
begin
        PositionValue := StartingPositionValue;
        DecimalValue := 0;

        while PositionValue <> 1 do begin

                PositionValue:= PositionValue/2;
                write('Enter a  bit 1 or 0  : ');
                readln(Bit);
                writeln;
                BaseValue := Bit*PositionValue;
                DecimalValue := DecimalValue+ BaseValue;

        end;
        writeln('The decimal eguivalent of the binary number is  :',DecimalValue:3:0);
        readln;
end.
```

3. State two things you have learnt from the program above.

4. Do a research on base 7 numbers, then modify the program above to read a 5-bit base 7 number. Convert it to its decimal value after carefully examining the logics program code above.