Practical Challenges in Programming

Workbook and Tracker

King Edward Vi School
Stratford-upon-Avon

Name: ………………………………………………………… Form:……………

Programming Tracker

Program	Focus	Designed	Coded	Tested	Python	VB.Net	Java	Other		
1. Hello World	Sequence									
2. Hello User	Sequence									
3. AreaCalc	Sequence, simple maths									
4. PerimeterCalc	Sequence, simple maths									
5. Bigger and Bigger	Selection									
6. Form Chooser	Selection									
7. VAT Calculator	Sequence, simple maths									
8. Vol and Surface Calc	Sequence, simple maths									
9. Mini-Calc	Selection, simple maths									
10. Cafe Adder	Selection, simple maths									
11, Five-A-Day	Iteration									
12. Countdown	Iteration									
13. Times-tabler	Iteration									
14. Squares and Cubes	Iteration, simple maths									
15. Fibonacci	Iteration, moderate maths									
16. FizzBuzz	Iteration, moderate maths									
17. Reversal	Arrays									
18. Bingo Bingo	Arrays									
19. Sort it Out!	Arrays									
20. Student Average	Arrays, simple maths									
21. Username	Strings									
22. Back to Front	Strings									
23. Word Count	Strings									
24. Palindromes	Strings									
25. Vowel Counter	Strings, arrays									
26. Anagrams	Strings, arrays									
27. Code Words 1	Strings									
28. Code Words 2	Strings									
29. Decoder	Strings									
30. Initialise	Strings									

Contents

Practical Challenges Part 1: Sequence and Selection

Practical Challenges Part 2: Iteration and Arrays

Practical Challenges Part 3: Strings and Things

Practical Challenges Part 4: The Big Ones

Note to teachers

This booklet covers most of the key practical techniques and skills required for both GCSE and AS level Computing. Please feel free to print, photocopy, distribute and use in whatever manner you wish. The layout is a little cramped at times, but has been designed to allow challenges to be printed off separately or as back-to-back worksheets.

The tasks have been adapted from a wide variety of sources and were originally written to help students learn the VB.Net language in preparation for OCR GCSE and AS level courses. The current version aims to be reasonably language agnostic. Students are encouraged to think, plan and design before developing their solutions and to test and evaluate afterwards. A number of theory questions and tasks are incorporated into the booklet, as well as opportunities to present Showcases, which may be used for formal assessment or as preparation for Controlled Assessment tasks. Students may use the tracker and target sheets on the inside covers to assess their progress and inform their next choice of task.

The tasks in each section are presented roughly in order of complexity/difficulty and cover the following key aspects of designing and creating a programmed solution to a problem:

Variables
Constants
Datatypes
Arrays
Loops
Conditions/Selection
String manipulation

The final section gives some suggestions for slightly longer programs, with the student expected to become increasingly independent and able to plan, design and create their programs with minimal supervision. Many of the final programs can be extended or turned into full games/apps.

Additional sections covering file handling, modules, recursion and graphics will be added to the next version.

Corrections, comments and suggestions for improvements are always welcomed via the CAS forums.

Richard Barfoot

Note to Students

The challenges in this booklet have been put together to help you plan, design, create and then test a range of programs. When you start your programming journey, experience and practice are absolutely vital. Knowing when and how to use certain features of a programming language, such as conditions and loops, are often difficult at first. Thinking about your program BEFORE you create any code is highly recommended, and indeed, vital at GCSE and AS/A2 Level for the higher grades. You can use the grids on the inside covers to help monitor your progress and track the programming skills you have begun to master.

BEFORE creating a program you should:

- Identify the Inputs, Processing and Outputs
- Identify key variables and other data structures
- Design a labelled interface (for non-console apps)
- Think about the steps the program should execute (using flowcharts and/or pseudocode)
- Draw up a test plan

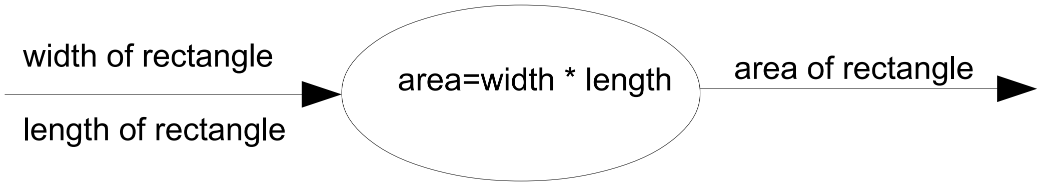
After you have created your program, it should be thoroughly tested using a test plan that covers valid, invalid and extreme data. Only when all of the challenge is complete should you mark it off on your tracker grids.

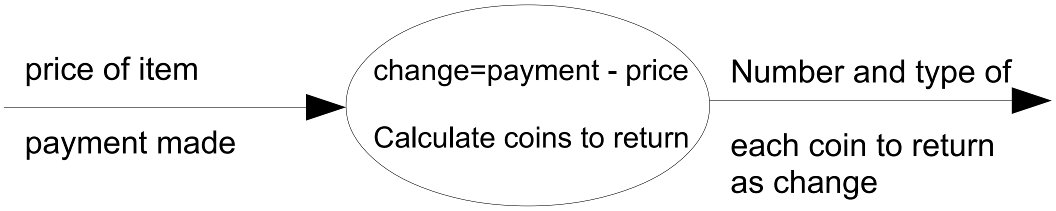
The challenges were originally written with the VB.Net language in mind. The wording of some of the written questions may not be 100% familiar to users of other languages, although similar features and concepts will be available in their language of choice.

I->P->O Diagram

This simple diagram helps you focus on what goes ***into*** the program and what should come ***out***. Inputs and Outputs should be stated as clearly and accurately as possible and should help you start to think about the variables needed for the program. The processing may be expressed simply at this stage.

Examples:




price of item
payment made
change=payment - price
Calculate coins to return
Number and type of each coin to return as change

Variables and Other Data Structure

Nearly all programs will need to store and manipulate data in some way. You should work out the main requirements before you create the program. Think about the structures needed to store the inputs and outputs of the program. Think about the actual processing involved – does it need to be done in stages and if so, what extra data structures are needed? Would an array or list be appropriate?

BEFORE creating a program, you should think about the following for each data structure needed: name; datatype; typical value; minimum/maximum values (validation).

Flowcharts and Pseudocode

Drawing a flowchart or diagram should help you think about the various stages needed in the program: What are they? In what order should they be executed (sequence)? Which sections involve making decisions (selection)? Which sections involve repeating instructions (iteration)?

Pseudocode should help you get to grips with the hardest part of the program: the processing. How exactly are you going to turn those inputs into the expected outputs? What needs to be done? Is it a mathematical formula? Do you need to manipulate strings?

A good piece of pseudocode focuses on the logic and helps solve the difficult parts of the problem so that when you come to actually writing your code, most of the thinking has already been done. Good pseudocode will also help you work out where you might go wrong or what problems/traps might lay in store (eg what happens if an input is left blank?)

Avoid Pointless Pseudocode

```
Input number1
Input number2
Input number3
Input number4
Input number5
Output average
```

Version 1 doesn't really help...

```
Input 5 numbers
average=(sum of the five numbers) / 5
Output average
```

Although shorter, Version 2 is a bit more helpful

```
numberOfItems=5
total=0
average=0
for c=1 to numberOfItems
    Output “Please enter a number”
    Input number
    total=total+number
next
average=total / numberOfItems
Output “The average is “ & average
```

In Version3, nearly all the “thinking” has been done

Test Plan

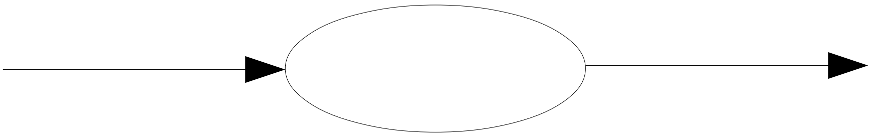
How will you know if your program works correctly? How will you know what crashes/kills your program?

Create a test plan **BEFORE** you write the code. You should be able to predict the output created by a set of inputs. Try to design tests that cover valid, invalid and extreme inputs. A good test will highlight a problem/bug; there's no point in 10 “it works ok” tests.

In some of the challenges, a few starting tests have been given but feel free to add more!

Part 1: Sequence and Selection

1. Hello World
Design and implement a program that displays the message “Hello World” when a button is pressed



Pseudocode:

…..

…...

Explain these terms:

Form:...

Control:..

Button:...

Label:...

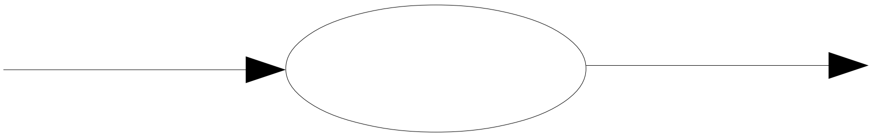
Event:...

Implement your program!

Testing
How do you know whether your program works?

2. Hello User

Design and implement a program that allows the user to enter his/her name. When a button is clicked, the program displays “Hello *user*”, where *user* is the name entered.



Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Pseudocode:

…..

....

....

Explain these terms:

Textbox:...

Property:...

String:...

Concatenation:...

Implement your program!

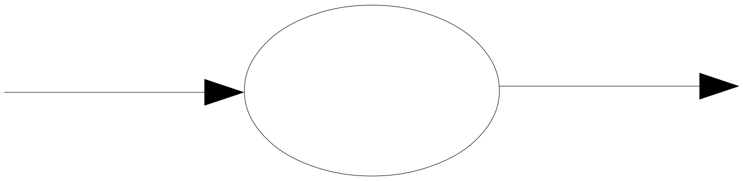
What is the difference between the Name property of an object/control and its Text property?

…...

…...

3. AreaCalc

Design and implement a program that allows the user to enter 2 numbers representing the width and length of a rectangle. The program calculates and displays the area of the rectangle.



Flowchart/Pseudocode

…...

…...

…...

…...

…...

…...

…...

…...

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Output	Actual Output	Comments
2, 3			
100, 100			
4.2, 1.8			
1m, 3			

Implement your program!

What is the difference between an integer and a real number?

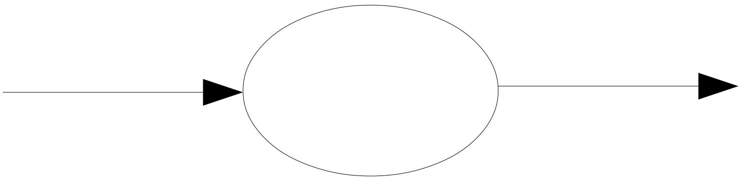
…...

Testing

Use the table above to identify any issues with your program

4. PerimeterCalc

Design and implement a program that allows the user to enter 2 numbers representing the width and length of a rectangle. The program calculates and displays the perimeter of the rectangle.



Flowchart/Pseudocode

…...

…...

…...

…...

…...

…...

…...

…...

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Output	Actual Output	Comments
2, 3			
100, 100			
4.2, 1.8			
1m, 3			
1, 3m			

Implement your program!

Testing

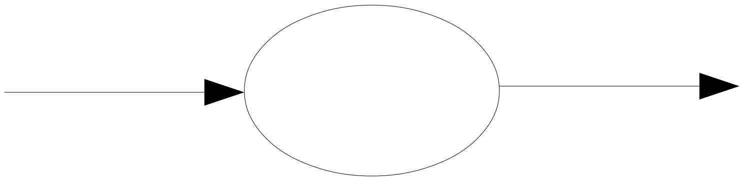
Use the table above to identify any issues with your program

What is the purpose of the 4th and 5th tests in the test plan?

…..

5. Bigger and Bigger

Design and implement a program that allows the user to enter 3 numbers in any order. The program displays the largest number.



Flowchart/Pseudocode

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Output	Actual Output	Comments
2, 9, 3			
100, 4, 47			
4.2, 1.8, 5			
1, 0, -1			
5, 8, 5			

Implement your program!

Testing

Use the table above to identify any issues with your program

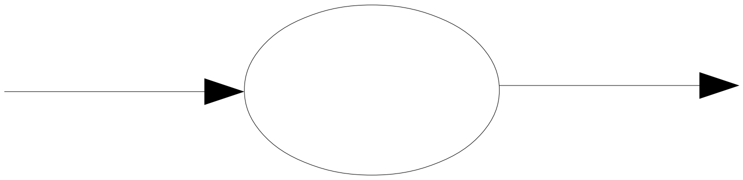
What is the purpose of the 5th test in the test plan?

………

6. Form Chooser

Design and implement a program that allows the user to enter his/her name and form. When a button is clicked, the program greets the user by name and tells him/her which room to go to for registration.

Eg Inputs: ***Ben, 9T***
Output: ***Hello Ben, please go to Room 16***



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines. Include **valid** and **invalid** data.

Input	Expected Output	Actual Output	Comments

Explain these terms:

Selection:...

Alternative Route:...

Error Checking:...

Valid data:...

Invalid data:..

Validation:...

Implement your program!

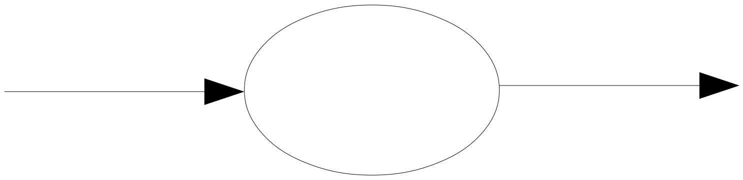
Hint:

You may choose to use ***If...Then...End If*** statements
You may choose to use a ***Select...Case...End Select*** statement

add a printout of your code here

7. VAT Calculator

Design and implement a program that allows the user to enter the price of an item. The program calculates the amount of VAT to be paid at 20%. The VAT amount and total price are then displayed.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Outputs	Actual Outputs	Comments
0.50			
50.00			
35.75			
£10.00			
Ten pounds			

Explain these terms:

Variable:..

Initialise:..

Constant:..

Integer:..

Floating Point:..

Double:..

Implement your program!

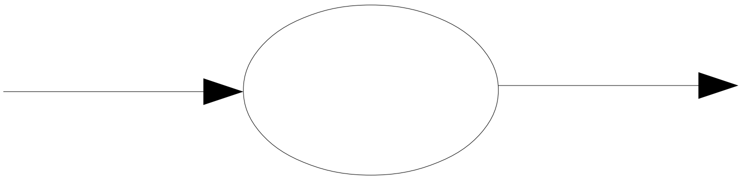
Hint:

You may need to use ***val()*** to get the value of the price entered

add a printout of your code here

8. Volume and Surface Calc

Design and implement a program that allows the user to enter 3 numbers representing the height, width and length of a cuboid. The program calculates and displays the volume and total surface area of the cuboid.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Outputs	Actual Outputs	Comments
2, 2, 2			
3, 3, 3			
2.5, 3, 4			
4, 6, 8			
4, 8, 8			
3, 0, 3			

Implement your program!

Testing

Use the table above to identify any issues with your program

What is the purpose of the 3th and 6th tests in the test plan?

…...

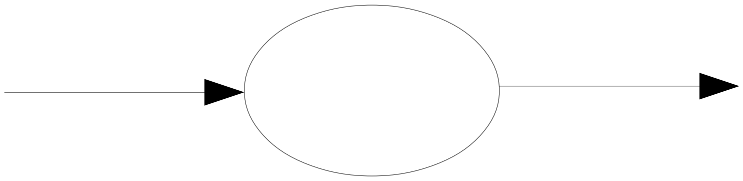
…...

add a printout of your code here

9. Mini-Calc

Design and implement a program that allows the user to enter 2 numbers and choose a mathematical operation (+ - * /). The program calculates the result and displays the answer as a full equation (eg the user enters **4** and **10** and chooses **+**. The program displays **4 + 10 = 14**

NB: the interface does NOT have to look like a calculator!



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
4, 10, +			
4, 10, -			
4, 10, *			
40, 10, /			
25.5, 5, /			
100000, 10000, *			
10,0, /			
10, , +			
Ten, 4, +			

Implement your program!

Hint:
You may need to use ***val()*** to get the value of the numbers entered
You may choose to use a drop-down list for the 4 mathematical operations.

Testing

Use the table above to identify any issues with your program

Why do the first 4 tests use the same numbers?

…...

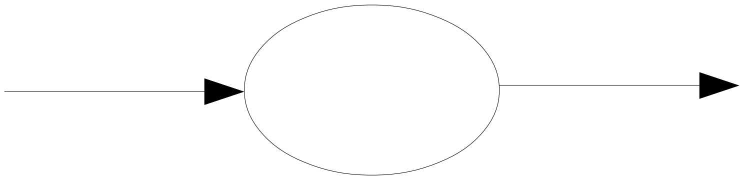
What is the purpose of the 6th test in the test plan?

…...

add a printout of your code here

10. Café Adder

Design and implement a program that allows the user to create the bill for a café. The user can enter the name and price for up to 5 items and how many were ordered. If the order is to take-out, 20% VAT is added. If the customer is a pensioner, a 10% discount is applied. The program displays an appropriate bill/receipt..



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
Coffee, £2, 2 Cake £1.50, 2			
Coffee, £2, 2 Cake £1.50, 2, Take-out			
Coffee, £2, 2 Cake £1.50, 2, Pensioner			

Implement your program!

Hint:

It is very easy to over-complicate this one. Don't!
You could choose drop-down boxes to limit the variety of items and their prices
You might choose to use check-boxes for Take-Away and Pensioner inputs
You might choose to use an array or you might not!

Testing

Use the table above to identify any issues with your program

The test data has values such as “£2” and “£1.50”? Why are these not correct in reality?

…..

Programming Showcase

A Programming Showcase is a formal document showing how you progressed through all of the stages in the development process. It will use some of the design, development and testing work you have completed in this booklet as well as a detailed narrative of ***what*** you did at each stage, and, more importantly, ***why*** and ***how*** you did it. It will also include a full ***evaluation*** of the ***final*** version of your program and incorporate feedback and thoughts from other people (eg members of your class).

The showcase will be marked using the exam board's guidelines for Controlled Assessment or Coursework Projects.

The Showcase should cover the following main points:

Aim

Description of program's main purposes(s). You could paraphrase the task's instructions.

Design and Planning

I->P->O diagram
Flowchart
Pseudocode
Variables/data structures needed (and validation)
For the highest grades, remember to discuss ***why*** you are using a particular approach

Test Plan

Detailed testing strategy – how will you know if your program works?
Detailed testing table – what data will you use to test your program?
Valid, invalid and extreme data

Development

Annotated screenshots showing stages of development
Discussion of any problems you encountered and how you fixed them (or didn't!)
Explicit, detailed discussion of changes made to your code following alpha testing
Full code listing for the final program
Code should be commented, indented, use sensible variable names etc

Testing

Completed test plan with appropriate comments and evidence (screenshots?)

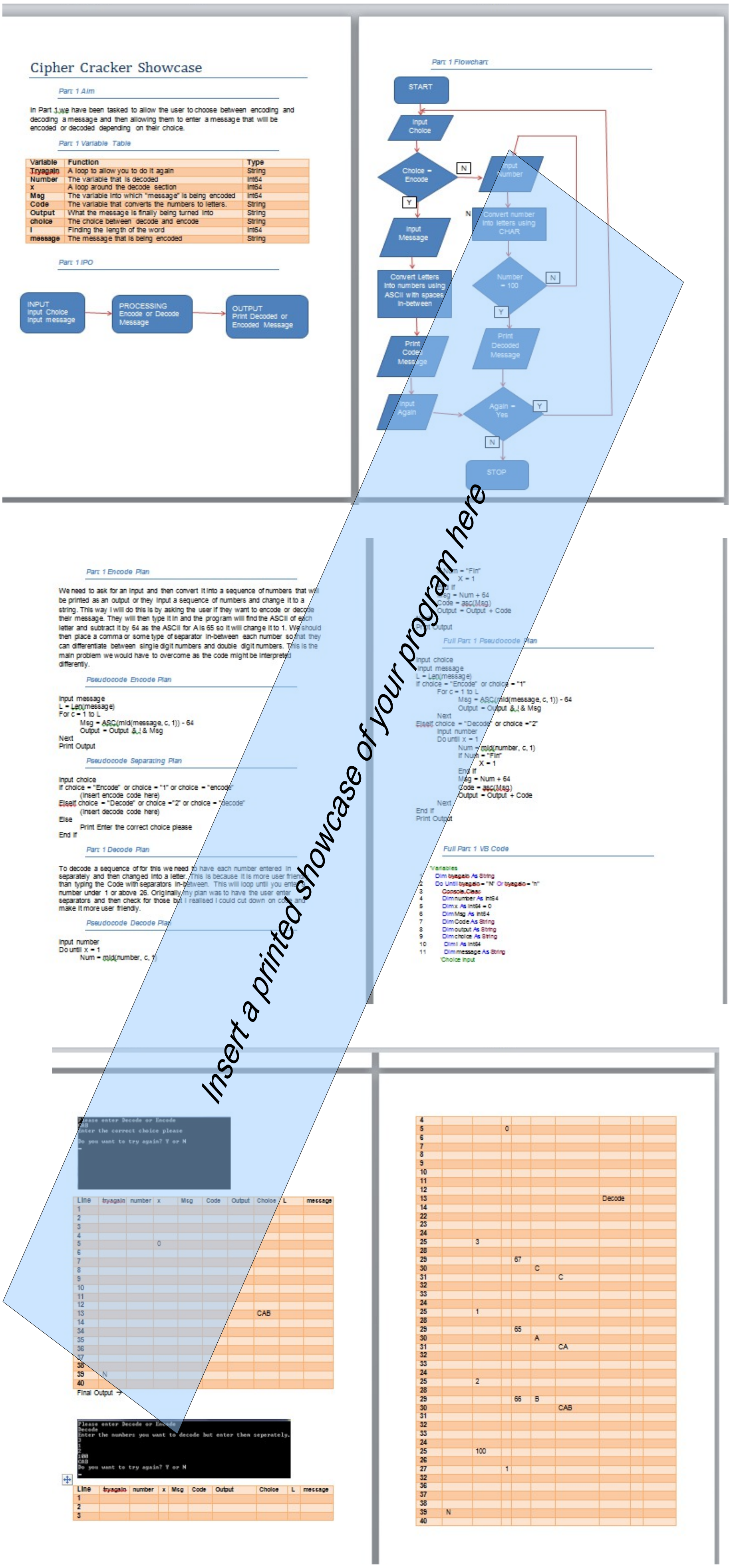
Evaluation

Discuss good/bad aspects
Feedback from others
Suggestions for further improvement
Note how the following evaluation paragraphs each cover a **Fact**, an **Opinion** and an **Improvement**
(F->O->I)

The interface for the program uses drop-down lists so the user can choose the items bought. This is good because it prevents the user from entering invalid spellings or non-existent items. When the program first loads up, the drop-down lists always shows "coffee". I could set them to be blank so the user doesn't have to change them to blank if fewer than 5 items are bought.

The program correctly works out the student's average score in most cases. However, the maximum score for each test is supposed to be 20 and it is possible to enter higher scores than this. I could add some error-handling to prevent the user from entering invalid numbers, eg:

```
repeat
      input score
until score>-1 and score<21
```



Insert a printed showcase of your program here

Cipher Cracker Showcase

Part 1 Aim

In Part 1 we have been tasked to allow the user to choose between encoding and decoding a message and then allowing them to enter a message that will be encoded or decoded depending on their choice.

Part 1 Variable Table

Variable	Function	Type
Tryagain	A loop to allow you to do it again	String
Number	The variable that is decoded	Int64
x	A loop around the decode section	Int64
Msg	The variable into which "message" is being encoded	Int64
Code	The variable that converts the numbers to letters.	String
Output	What the message is finally being turned into	String
choice	The choice between decode and encode	String
l	Finding the length of the word	Int64
message	The message that is being encoded	String

Part 1 IPO

Part 1 Flowchart

Part 1 Encode Plan

We need to ask for an input and then convert it into a sequence of numbers that will be printed as an output or they input a sequence of numbers and change it to a string. This way I will do this is by asking the user if they want to encode or decode their message. They will then type it in and the program will find the ASCII of each letter and subtract it by 64 as the ASCII for A is 65 so it will change it to 1. We should then place a comma or some type of separator in-between each number so that they can differentiate between single digit numbers and double digit numbers. This is the main problem we would have to overcome as the code might be interpreted differently.

Pseudocode Encode Plan

```
Input message
L = Len(message)
For c = 1 to L
        Msg = ASC(mid(message, c, 1)) - 64
        Output = Output & ! & Msg
Next
Print Output
```

Pseudocode Separating Plan

```
Input choice
If choice = "Encode" or choice = "1" or choice = "encode"
        (Insert encode code here)
Elseif choice = "Decode" or choice ="2" or choice = "decode"
        (Insert decode code here)
Else
        Print Enter the correct choice please
End If
```

Part 1 Decode Plan

To decode a sequence of for this we need to have each number entered in separately and then changed into a letter. This is because it is more user friendly than typing the Code with separators in-between. This will loop until you enter a number under 1 or above 26. Originally my plan was to have the user enter separators and then check for those but I realised I could cut down on code and make it more user friendly.

Pseudocode Decode Plan

```
Input number
Do until x = 1
        Num = mid(number, c, 1)
        If Num = "Fin"
                X = 1
        End If
        Msg = Num + 64
        Code = asc(Msg)
        Output = Output + Code
Next
Print Output
```

Full Part 1 Pseudocode Plan

```
Input choice
Input message
L = Len(message)
If choice = "Encode" or choice = "1"
        For c = 1 to L
                Msg = ASC(mid(message, c, 1)) - 64
                Output = Output & ! & Msg
        Next
Elseif choice = "Decode" or choice ="2"
        Input number
        Do until x = 1
                Num = mid(number, c, 1)
                If Num = "Fin"
                        X = 1
                End If
                Msg = Num + 64
                Code = asc(Msg)
                Output = Output + Code
        Next
End If
Print Output
```

Full Part 1 VB Code

```
     'Variables
1      Dim tryagain As String
2      Do Until tryagain = "N" Or tryagain = "n"
3          Console.Clear
4          Dim number As Int64
5          Dim x As Int64 = 0
6          Dim Msg As Int64
7          Dim Code As String
8          Dim output As String
9          Dim choice As String
10         Dim l As Int64
11         Dim message As String
           'Choice Input
```

```
Please enter Decode or Encode
CAB
Enter the correct choice please

Do you want to try again? Y or N
_
```

Line	tryagain	number	x	Msg	Code	Output	Choice	L	message
1									
2									
3									
4									
5			0						
6									
7									
8									
9									
10									
11									
12									
13							CAB		
14									
34									
35									
36									
37									
38									
39	N								
40									

Final Output →

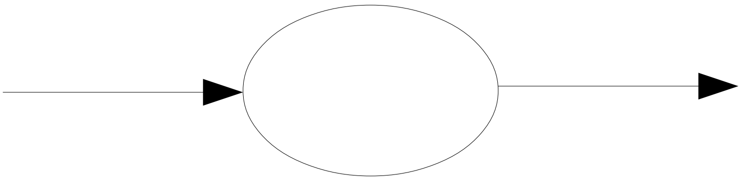
```
Please enter Decode or Encode
Decode
Enter the numbers you want to decode but enter them seperately.
3
1
2
100
CAB
Do you want to try again? Y or N
_
```

Line	tryagain	number	x	Msg	Code	Output	Choice	L	message
1									
2									
3									
4									
5			0						
6									
7									
8									
9									
10									
11									
12									
13							Decode		
14									
22									
23									
24									
25		3							
28									
29				67					
30					C				
31						C			
32									
33									
24									
25		1							
28									
29				65					
30					A				
31						CA			
32									
33									
24									
25		2							
28									
29				66	B				
30						CAB			
31									
32									
33									
24									
25		100							
26									
27			1						
32									
36									
37									
38									
39	N								
40									

Part 2: Iteration and Arrays

11. Five-A-Day

Design and implement a program that allows the user to enter a positive number. The program displays all of the numbers divisible by 5 up to the number entered. Eg the user enters 30, the program displays 5, 10, 15, 20, 25, 30



Flowchart/Pseudocode

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Output	Actual Output	Comments
30			
100000			
86			
0			
-27			

Implement your program!

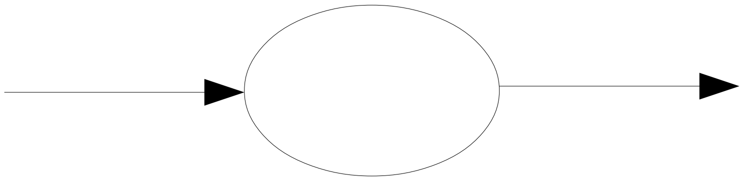
Testing

Use the table above to identify any issues with your program
What is the purpose of the 4th and 5th tests in the test plan?

………

12. Countdown!

Design and implement a program that allows the user to enter a positive number. The program displays a countdown from the number to 0. Eg the user enters 10 and the program displays 10, 9, 8, 7, 6, 5 4, 3, 2, 1, 0



Flowchart/Pseudocode

…...

…...

…...

…...

…...

…...

…...

…...

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Output	Actual Output	Comments
10			
100			
1000000			
0			
-5			

Implement your program!

Testing

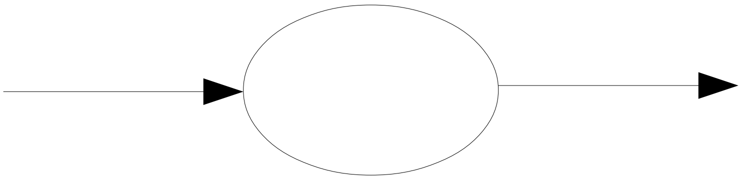
Use the table above to identify any issues with your program

What is the purpose of the 5th test in the test plan?

…...

13. Times-Tabler

Design and write a program that allows the user to choose a number (1-12). The program displays the times-table for the chosen number



Flowchart/Pseudocode

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Output	Actual Output	Comments
10			
7			
19			
0			
-5			

Implement your program!

Testing

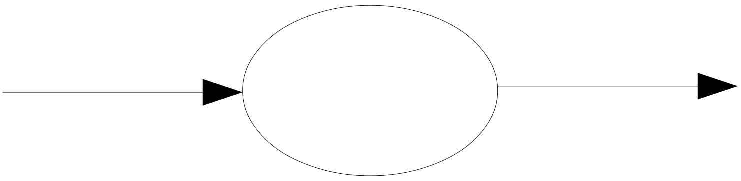
Use the table above to identify any issues with your program

What is the purpose of the 3rd, 4th and 5th tests in the test plan?

…..

14. Squares and Cubes

The squared number sequence starts: 1, 4, 9, 16, 25. The cubed number sequence starts: 1, 8, 27, 64. Design and write a program to display N numbers in the squared **or** cubed sequence according to user choice.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
squared, 5			
cubed, 5			
squared, 10			
cubed, 10			

Implement your program!

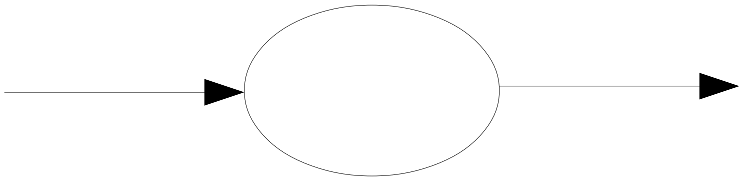
Testing

Use the table above to identify any issues with your program

add a printout of your code here

15. Fibonacci

The first few numbers in the Fibonacci sequence are: 0, 1, 1, 2, 3, 5, 8, 13, 21 …
Design and write a program to display N Fibonacci numbers, where N is entered by the user



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
3			
7			
9			
0			

Implement your program!

Testing

Use the table above to identify any issues with your program

Explain these terms:

Iteration:...

Loop:...

Stopping Condition:...

What is the difference between a FOR...NEXT loop and a REPEAT....UNTIL loop?

…...

What is the difference between a FOR...NEXT loop and a WHILE....DO loop?

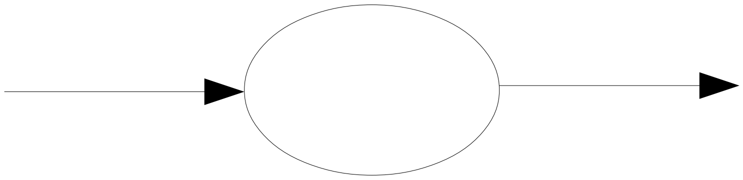
…...

What is the difference between a REPEAT...UNTIL loop and a REPEAT....UNTIL loop?

…...

16. FizzBuzz

Write a program to print out the numbers 1-*n*, where *n* is entered by the user. However, if a number is divisible by 3 (3, 6, 9 etc), it is replaced by “FIZZ!”. If a number is divisible by 5 (5, 10 etc) it is replaced by “BUZZ!” If a number is divisible by 3 **and** 5, it is replaced by “FIZZBUZZ!”



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
12			
36			
100			
0			
ten			

Implement your program!

Hint:

It is very easy to over-complicate this one. Don't!
Be careful with your interface – if there are 20 test scores to enter, do you really want 20 textboxes?
Do you want to use an array? Really?
Do you want to use iteration? A ***FOR...NEXT*** loop? A ***REPEAT...UNTIL*** loop? Really?

Testing

Use the table above to identify any issues with your program

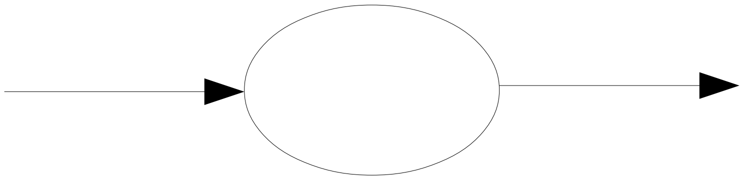
What is the purpose of the 4th and 5th tests?

…..

add a printout of your code here

17. Reversal

Write a program that allows the user to enter 5 numbers. The numbers are then displayed in reverse order.



Flowchart/Pseudocode

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected Output	Actual Output	Comments
5, 10, 12, 3, 9			

Implement your program!

Testing

Use the table above to identify any issues with your program

Explain these terms:

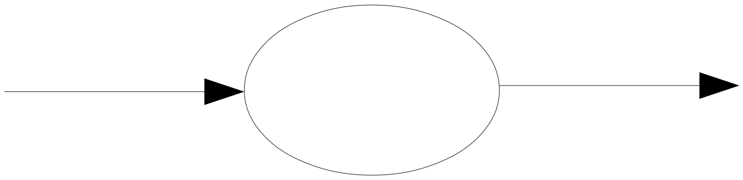
Array:………

Datatype:……

Index:………

18. Bingo Bingo

Write a program that allows the user to enter the Bingo Call for any number 1-90. The user can choose to have all the calls displayed. Eg ***1: Kelly's eye*** ***22: two little ducks*** etc



Flowchart/Pseudocode

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Input	Expected	Actual Output	Comments
1: Kelly's eye ***22: two little ducks***			

Implement your program!

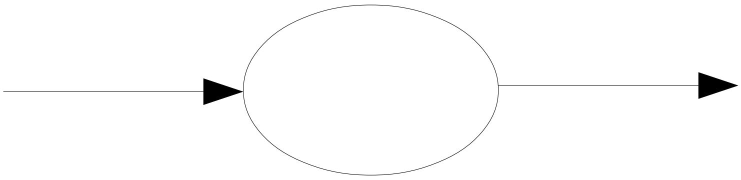
Hint:
You might want to make your program skip any numbers that haven't had a Bingo Call entered

Testing

Use the table above to identify any issues with your program

19. Sort It Out!

Write a program that allows the user to enter 5 words. The words are sorted and displayed in alphabetical order.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
and, but, car, egg, dust			
dust, egg, car, and, but			
egg, dust, car, but, and			
and, able, but, bit, bat			
and, but, car, car, dust			

Implement your program!

Hint:

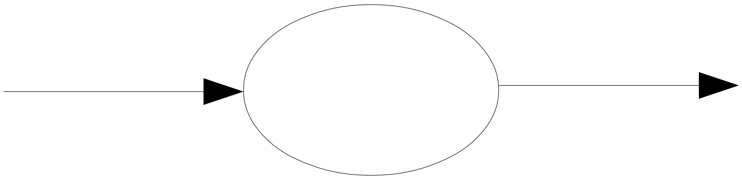
It is very easy to over-complicate this one. Don't!
Be careful with your interface – there are up to 30 test scores to enter, do you really want 30+ textboxes?
Do you want to use more than one array?
Do you want to use iteration? A ***FOR...NEXT*** loop? A ***REPEAT...UNTIL*** loop?

Testing

Use the table above to identify any issues with your program

20. Student Average

Design and implement a program that allows the user to enter the names of up to 10 students and the scores for each of the 3 tests they have taken (out of 20). The user can then enter a student number (1-10) and the program displays his/her name, test scores and average score.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments

Implement your program!

Hint:

It is very easy to over-complicate this one. Don't!
Be careful with your interface – there are up to 30 test scores to enter, do you really want 30+ textboxes?
Do you need to use more than one array?
Do you want to use iteration? A ***FOR...NEXT*** loop? A ***REPEAT...UNTIL*** loop?

Testing

Use the table above to identify any issues with your program

Programming Showcase

A Programming Showcase is a formal document showing how you progressed through all of the stages in the development process. It will use some of the design, development and testing work you have completed in this booklet as well as a detailed narrative of ***what*** you did at each stage, and, more importantly, ***why*** and ***how*** you did it. It will also include a full ***evaluation*** of the ***final*** version of your program and incorporate feedback and thoughts from other people (eg members of your class).

The showcase will be marked using the exam board's guidelines for Controlled Assessment or Coursework Projects.

The Showcase should cover the following main points:

Aim

Description of program's main purposes(s). You could paraphrase the task's instructions.

Design and Planning

I->P->O diagram
Flowchart
Pseudocode
Variables/data structures needed (and validation)
For the highest grades, remember to discuss ***why*** you are using a particular approach

Test Plan

Detailed testing strategy – how will you know if your program works?
Detailed testing table – what data will you use to test your program?
Valid, invalid and extreme data

Development

Annotated screenshots showing stages of development
Discussion of any problems you encountered and how you fixed them (or didn't!)
Explicit, detailed discussion of changes made to your code following alpha testing
Full code listing for the final program
Code should be commented, indented, use sensible variable names etc

Testing

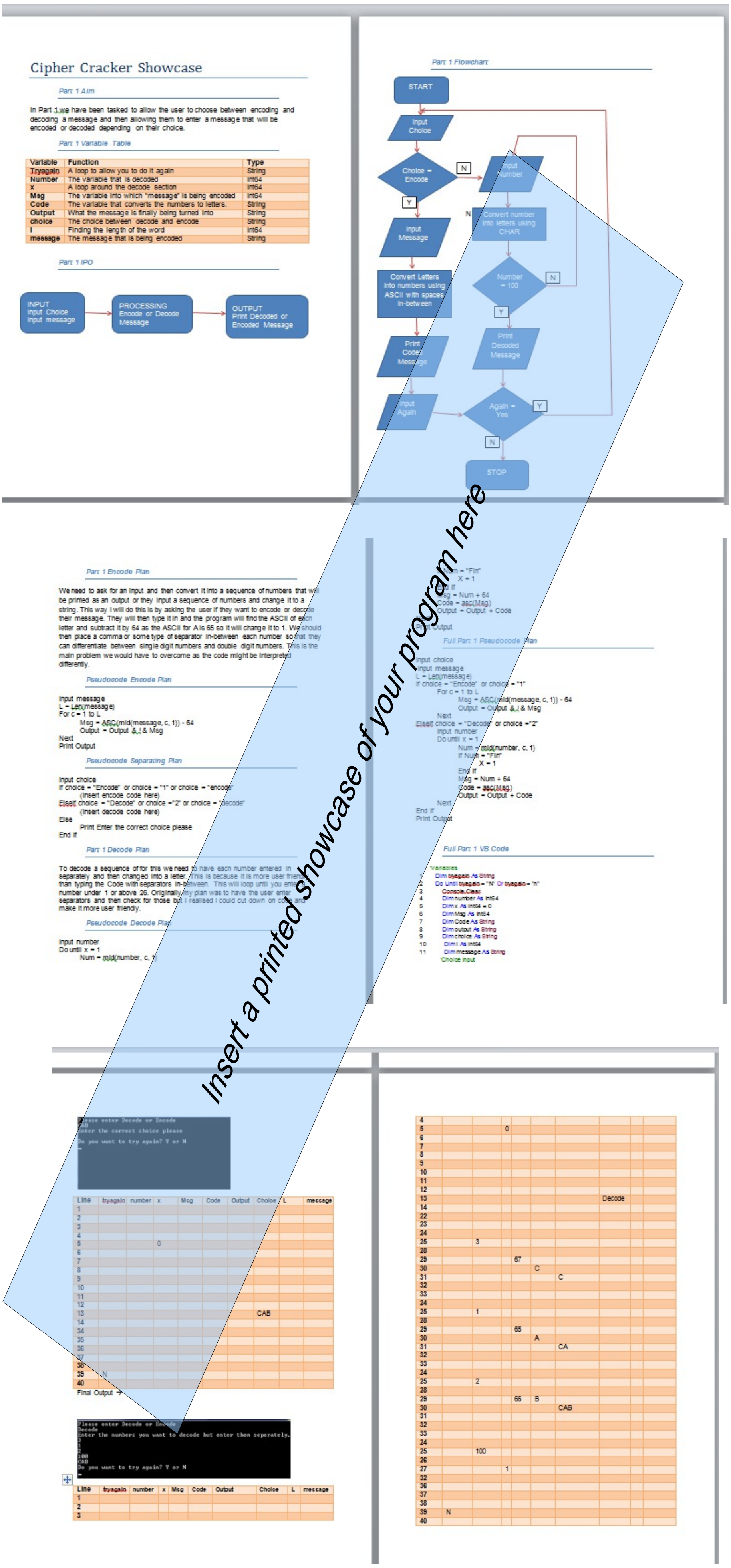
Completed test plan with appropriate comments and evidence (screenshots?)

Evaluation

Discuss good/bad aspects
Feedback from others
Suggestions for further improvement
Note how the following evaluation paragraphs each cover a **Fact**, an **Opinion** and an **Improvement**
(F->O->I)

> *The interface for the program uses drop-down lists so the user can choose the items bought. This is good because it prevents the user from entering invalid spellings or non-existent items. When the program first loads up, the drop-down list is blank. I could improve this be setting it to show a common item, such as coffee.*
>
> *The program correctly works out the student's average score in most cases. However, the maximum score for each test is supposed to be 20 and it is possible to enter higher scores than this. I could add some error-handling to prevent the user from entering invalid numbers, eg:*

```
repeat
    input score
until score>-1 and score<21
```



Insert a printed showcase of your program here

Cipher Cracker Showcase

Part 1 Aim

In Part 1 we have been tasked to allow the user to choose between encoding and decoding a message and then allowing them to enter a message that will be encoded or decoded depending on their choice.

Part 1 Variable Table

Variable	Function	Type
Tryagain	A loop to allow you to do it again	String
Number	The variable that is decoded	Int64
x	A loop around the decode section	Int64
Msg	The variable into which "message" is being encoded	Int64
Code	The variable that converts the numbers to letters.	String
Output	What the message is finally being turned into	String
choice	The choice between decode and encode	String
l	Finding the length of the word	Int64
message	The message that is being encoded	String

Part 1 IPO

INPUT
Input Choice
Input message

PROCESSING
Encode or Decode
Message

OUTPUT
Print Decoded or
Encoded Message

Part 1 Flowchart

Part 1 Encode Plan

We need to ask for an input and then convert it into a sequence of numbers that will be printed as an output or they input a sequence of numbers and change it to a string. This way I will do this is by asking the user if they want to encode or decode their message. They will then type it in and the program will find the ASCII of each letter and subtract it by 64 as the ASCII for A is 65 so it will change it to 1. We should then place a comma or some type of separator in-between each number so that they can differentiate between single digit numbers and double digit numbers. This is the main problem we would have to overcome as the code might be interpreted differently.

Pseudocode Encode Plan

```
Input message
L = Len(message)
For c = 1 to L
        Msg = ASC(mid(message, c, 1)) - 64
        Output = Output &.! & Msg
Next
Print Output
```

Pseudocode Separating Plan

```
Input choice
If choice = "Encode" or choice = "1" or choice = "encode"
        (insert encode code here)
Elseif choice = "Decode" or choice ="2" or choice = "decode"
        (insert decode code here)
Else
        Print Enter the correct choice please
End If
```

Part 1 Decode Plan

To decode a sequence of for this we need to have each number entered in separately and then changed into a letter. This is because it is more user friendly than typing the Code with separators in-between. This will loop until you enter a number under 1 or above 26. Originally my plan was to have the user enter separators and then check for those but I realised I could cut down on code and make it more user friendly.

Pseudocode Decode Plan

```
Input number
Do until x = 1
        Num = mid(number, c, 1)
        If Num = "Fin"
                X = 1
        End If
        Msg = Num + 64
        Code = asc(Msg)
        Output = Output + Code
Next
Print Output
```

Full Part 1 Pseudocode Plan

```
Input choice
Input message
L = Len(message)
If choice = "Encode" or choice = "1"
        For c = 1 to L
                Msg = ASC(mid(message, c, 1)) - 64
                Output = Output &.! & Msg
        Next
Elseif choice = "Decode" or choice ="2"
        Input number
        Do until x = 1
                Num = mid(number, c, 1)
                If Num = "Fin"
                        X = 1
                End If
                Msg = Num + 64
                Code = asc(Msg)
                Output = Output + Code
        Next
End If
Print Output
```

Full Part 1 VB Code

```
'Variables
    Dim tryagain As String
    Do Until tryagain = "N" Or tryagain = "n"
        Console.Clear
        Dim number As Int64
        Dim x As Int64 = 0
        Dim Msg As Int64
        Dim Code As String
        Dim output As String
        Dim choice As String
        Dim l As Int64
        Dim message As String
       'Choice Input
```

```
Please enter Decode or Encode
CAB
Enter the correct choice please

Do you want to try again? Y or N
_
```

Line	tryagain	number	x	Msg	Code	Output	Choice	L	message
1									
2									
3									
4									
5			0						
6									
7									
8									
9									
10									
11									
12									
13							CAB		
14									
34									
35									
36									
37									
38									
39	N								
40									

Final Output →

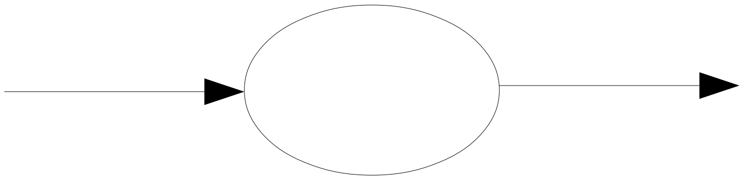
```
Please enter Decode or Encode
Decode
Enter the numbers you want to decode but enter them seperately.
3
1
2
100
CAB
Do you want to try again? Y or N
_
```

Line	tryagain	number	x	Msg	Code	Output	Choice	L	message
1									
2									
3									
4									
5			0						
6									
7									
8									
9									
10									
11									
12									
13							Decode		
14									
22									
23									
24									
25		3							
28									
29				67					
30					C				
31						C			
32									
33									
24									
25		1							
28									
29				65					
30					A				
31						CA			
32									
33									
24									
25		2							
28									
29				66	B				
30						CAB			
31									
32									
33									
24									
25		100							
26									
27			1						
32									
36									
37									
38									
39	N								
40									

Part 3: String and Things

21. Username
Write a program that inputs the user's first name and surname. It then generates and prints a username following the convention SURNAME.INITIAL

eg entering “John” and “Smith” gives the username “**Smith.J**”



Flowchart/Pseudocode

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

…………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
Alan Jones			
Mark Freeman			
Andy Barnes-Nobles			
Frank			
(blank)			

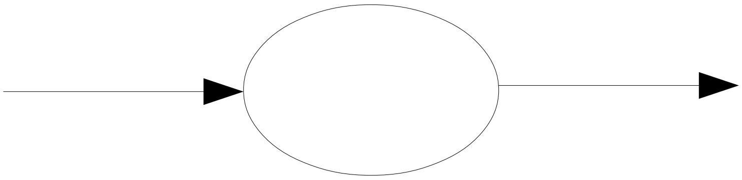
Implement your program!

Testing

Use the table above to identify any issues with your program

22. Back to Front

Design and write a program that reverses a piece of text entered by the user.



Flowchart/Pseudocode

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

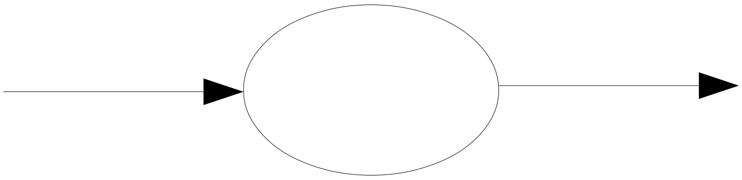
Inputs	Expected Output	Actual Outputs	Comments
computing			
Is cool			

Implement your program!

Testing

Use the table above to identify any issues with your program

23. Word Count
Design and write a program that counts the number of words in a passage of text entered by the user.



Flowchart/Pseudocode

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

……………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan
Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
Try this first			
Now try this one.			
And finally. Now for something, completely, different!			

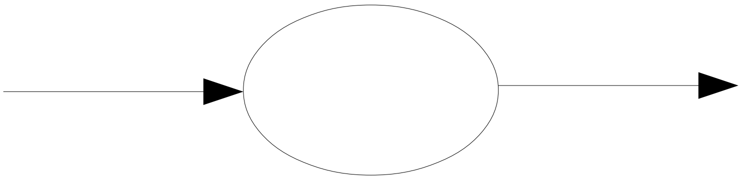
Implement your program!

Hint:
What is a word? Are you always “out by one”?

Testing
Use the table above to identify any issues with your program

24. Palindromes

A palindrome is a word or phrase that is exactly the same when written forwards or backwards. Eg “noon”, “Dammit I'm mad!” (if you ignore the punctuation). Design and write a program that ascertains whether a word or phrase entered by the user is a palindrome.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
noon			
kayak			
dammit im mad			
Dammit! I'm mad!			
As I pee, sir, I see Pisa!			

Implement your program!

Testing

Use the table above to identify any issues with your program

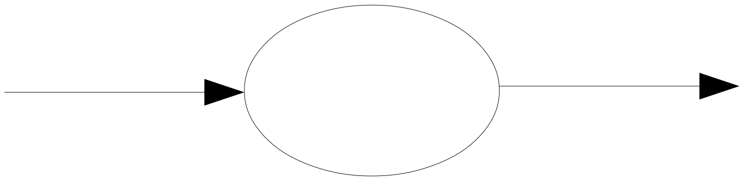
What is the purpose of the 4th test in the test plan?

…...

add a printout of your code here

25. Vowel Counter

Design and write a program that counts the number of each vowel in a passage of text entered by the user. Eg the user enters “**Computing is cool**” and the program displays: **a – 0, e – 0, i – 2, o – 3, u – 1**



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
Computing is cool			

Implement your program!

Hint:

Could you use an array to cut down on the amount of code?

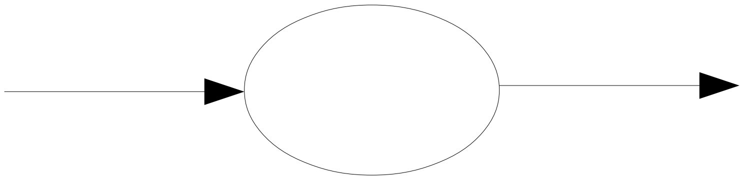
Testing

Use the table above to identify any issues with your program

add a printout of your code here

26. Anagrams

A word is an anagram of another word if it contains exactly the same letters, but in a different order. Design and create a program that ascertains whether 2 words are anagrams of each other. Eg “**reap**” is an anagram of “**pear**”, but not “peer”.



Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Flowchart

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
pear, reap			
was, saw			
saw, sore			
thin, hints			
maxi, maxim			

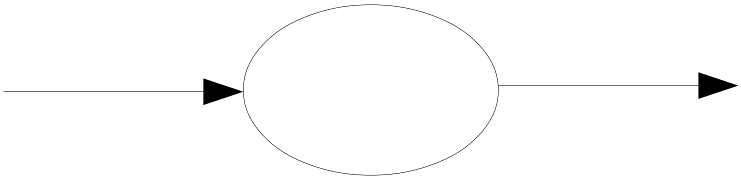
Implement your program!

Testing

Use the table above to identify any issues with your program

add a printout of your code here

27. Codewords 1
In a simple substitution code, the letter A is replaced by the number 1, B by the number 2 etc.
Design and write a program that allows the user to enter a short sentence, which is then encoded and displayed using the substitution rules.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
COMPUTING			
IS COOL			
TRU3			
dat			

Implement your program!

Hint:

Do you know about ASCII?

Testing

Use the table above to identify any issues with your program

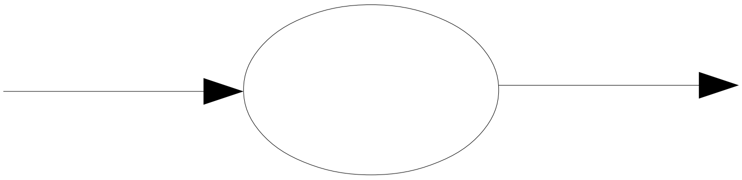
add a printout of your code here

28. CodeWords 2

Write a program that converts any entered word into a codeword using these rules:

- “egg” is inserted into the mid-point of any word of even-numbered length - eg “**good**” becomes “**goeggod**”

- “ga” is inserted either side of the mid-point of any word of odd numbered length eg “**puppy**” becomes “**pugapgapy**”



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
good			
puppy			

Implement your program!

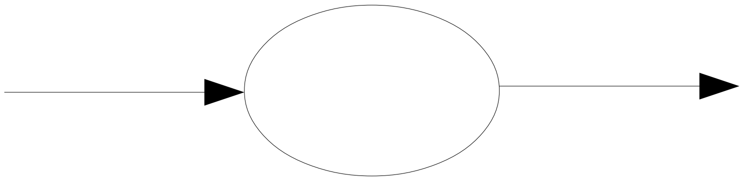
Testing

Use the table above to identify any issues with your program

add a printout of your code here

29. Initialise

Design and write a program that allows the user to enter a lengthy piece of text. The initial letter of each word is extracted and displayed in the hope of finding a secret message.
Eg the user enters **Today Harry indicated sadly “I shall frequently undergo nightmares**” and the program displays “**thisisfun**”



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
Begin untying Malcolm soon			
Today Harry indicated sadly “I shall frequently undergo nightmares”			

Implement your program!

Testing

Use the table above to identify any issues with your program

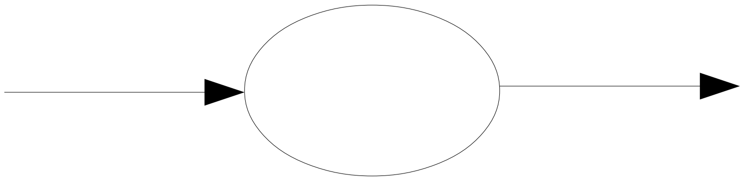
add a printout of your code here

30. Decoder

Design and write a program that decodes a message created using the program from Challenge 28: Code Words 2. The program should strip out any added “egg”s and “ga”s

Eg **goeggod** is turned back into **good**

Gaaga goeggod pugapgapy is turned back into **a good puppy**



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Add your own tests to the blank lines.

Inputs	Expected Output	Actual Outputs	Comments
goeggod			
pugapgapy			
Gaaga goeggod pugapgapy			

Implement your program!

Testing

Use the table above to identify any issues with your program

add a printout of your code here

Programming Showcase

A Programming Showcase is a formal document showing how you progressed through all of the stages in the development process. It will use some of the design, development and testing work you have completed in this booklet as well as a detailed narrative of ***what*** you did at each stage, and, more importantly, ***why*** and ***how*** you did it. It will also include a full ***evaluation*** of the ***final*** version of your program and incorporate feedback and thoughts from other people (eg members of your class).

The showcase will be marked using the exam board's guidelines for Controlled Assessment or Coursework Projects.

The Showcase should cover the following main points:

Aim

Description of program's main purposes(s). You could paraphrase the task's instructions.

Design and Planning

I->P->O diagram
Flowchart
Pseudocode
Variables/data structures needed (and validation)
For the highest grades, remember to discuss ***why*** you are using a particular approach

Test Plan

Detailed testing strategy – how will you know if your program works?
Detailed testing table – what data will you use to test your program?
Valid, invalid and extreme data

Development

Annotated screenshots showing stages of development
Discussion of any problems you encountered and how you fixed them (or didn't!)
Explicit, detailed discussion of changes made to your code following alpha testing
Full code listing for the final program
Code should be commented, indented, use sensible variable names etc

Testing

Completed test plan with appropriate comments and evidence (screenshots?)

Evaluation

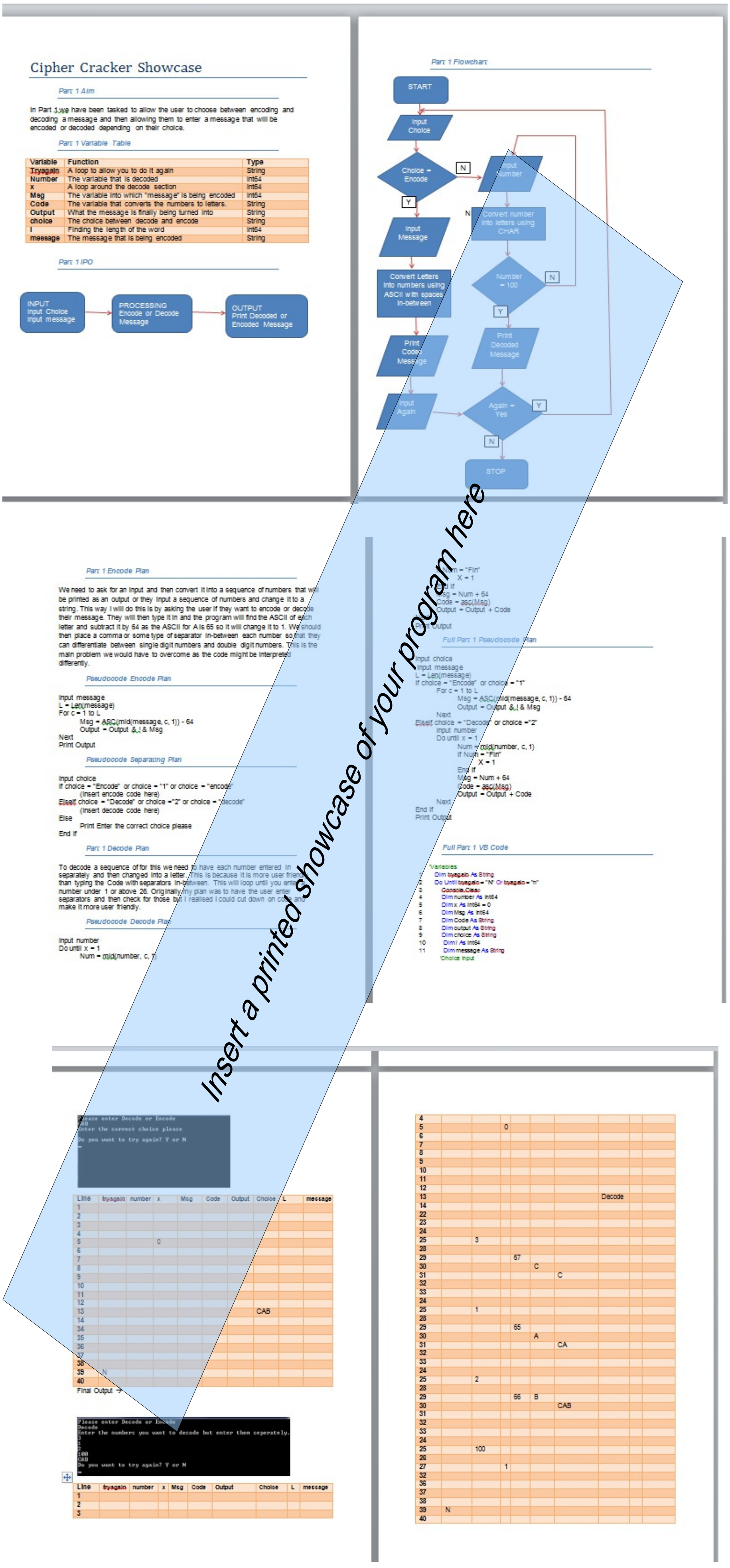
Discuss good/bad aspects
Feedback from others
Suggestions for further improvement
Note how the following evaluation paragraphs each cover a **Fact**, an **Opinion** and an **Improvement**
(F->O->I)

The interface for the program uses drop-down lists so the user can choose the items bought. This is good because it prevents the user from entering invalid spellings or non-existent items. When the program first loads up, the drop-down list is blank. I could improve this be setting it to show a common item, such as coffee.

The program correctly works out the student's average score in most cases. However, the maximum score for each test is supposed to be 20 and it is possible to enter higher scores than this. I could add some error-handling to prevent the user from entering invalid numbers, eg:

```
repeat
    input score
until score>-1 and score<21
```

Insert a printed showcase of your program here



Cipher Cracker Showcase

Part 1 Aim

In Part 1 we have been tasked to allow the user to choose between encoding and decoding a message and then allowing them to enter a message that will be encoded or decoded depending on their choice.

Part 1 Variable Table

Variable	Function	Type
Tryagain	A loop to allow you to do it again	String
Number	The variable that is decoded	Int64
x	A loop around the decode section	Int64
Msg	The variable into which "message" is being encoded	Int64
Code	The variable that converts the numbers to letters.	String
Output	What the message is finally being turned into	String
choice	The choice between decode and encode	String
l	Finding the length of the word	Int64
message	The message that is being encoded	String

Part 1 IPO

Part 1 Flowchart

Part 1 Encode Plan

We need to ask for an input and then convert it into a sequence of numbers that will be printed as an output or they input a sequence of numbers and change it to a string. This way I will do this is by asking the user if they want to encode or decode their message. They will then type it in and the program will find the ASCII of each letter and subtract it by 64 as the ASCII for A is 65 so it will change it to 1. We should then place a comma or some type of separator in-between each number so that they can differentiate between single digit numbers and double digit numbers. This is the main problem we would have to overcome as the code might be interpreted differently.

Pseudocode Encode Plan

```
Input message
L = Len(message)
For c = 1 to L
        Msg = ASC(mid(message, c, 1)) - 64
        Output = Output & "," & Msg
Next
Print Output
```

Pseudocode Separating Plan

```
Input choice
If choice = "Encode" or choice = "1" or choice = "encode"
        (insert encode code here)
ElseIf choice = "Decode" or choice ="2" or choice = "decode"
        (insert decode code here)
Else
        Print Enter the correct choice please
End If
```

Part 1 Decode Plan

To decode a sequence of for this we need to have each number entered in separately and then changed into a letter. This is because it is more user friendly than typing the Code with separators in-between. This will loop until you enter a number under 1 or above 26. Originally my plan was to have the user enter separators and then check for those but I realised I could cut down on code and make it more user friendly.

Pseudocode Decode Plan

```
Input number
Do until x = 1
        Num = mid(number, c, 1)
        If Num = "Fin"
                X = 1
        End If
        Msg = Num + 64
        Code = asc(Msg)
        Output = Output + Code
Next
Print Output
```

Full Part 1 Pseudocode Plan

```
Input choice
Input message
L = Len(message)
If choice = "Encode" or choice = "1"
        For c = 1 to L
                Msg = ASC(mid(message, c, 1)) - 64
                Output = Output & "," & Msg
        Next
ElseIf choice = "Decode" or choice ="2"
        Input number
        Do until x = 1
                Num = mid(number, c, 1)
                If Num = "Fin"
                        X = 1
                End If
                Msg = Num + 64
                Code = asc(Msg)
                Output = Output + Code
        Next
End If
Print Output
```

Full Part 1 VB Code

```
'Variables
  Dim tryagain As String
  Do Until tryagain = "N" Or tryagain = "n"
     Console.Clear
     Dim number As Int64
     Dim x As Int64 = 0
     Dim Msg As Int64
     Dim Code As String
     Dim output As String
     Dim choice As String
      Dim l As Int64
      Dim message As String
    'Choice Input
```

```
Please enter Decode or Encode
CAB
Enter the correct choice please

Do you want to try again? Y or N
_
```

Line	tryagain	number	x	Msg	Code	Output	Choice	L	message
1									
2									
3									
4									
5			0						
6									
7									
8									
9									
10									
11									
12									
13							CAB		
14									
34									
35									
36									
37									
38									
39	N								
40									

Final Output →

```
Please enter Decode or Encode
Decode
Enter the numbers you want to decode but enter them seperately.
3
1
2
100
CAB
Do you want to try again? Y or N
_
```

Line	tryagain	number	x	Msg	Code	Output	Choice	L	message
1									
2									
3									
4									
5			0						
6									
7									
8									
9									
10									
11									
12									
13							Decode		
14									
22									
23									
24									
25		3							
28									
29				67					
30					C				
31						C			
32									
33									
24									
25		1							
28									
29				65					
30					A				
31						CA			
32									
33									
24									
25		2							
28									
29				66	B				
30						CAB			
31									
32									
33									
24									
25		100							
26									
27			1						
32									
36									
37									
38									
39	N								
40									

Part 4: The Big Ones

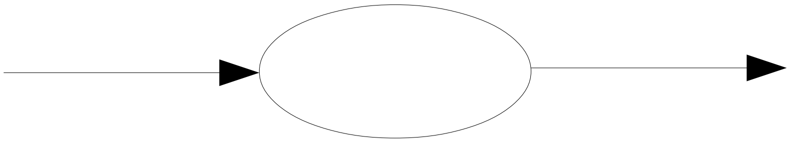
31. Carpet Quoter

A carpet store needs a system to create quotes for customers' queries. The program should allow the user to enter:

- Width and length of room in metres
- Cost of carpet per square metre
- Optional fixed delivery cost (£20)
- Optional fitting cost of £5 per square metre
- Optional 10% sales discount (on carpet only, not on delivery or fitting)

The program should display:

- Sub total, the amount of VAT charged (at 20%) and the Final Total



Flowchart

Pseudocode (focus on the decisions needed):

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. Remember to include valid, invalid and extreme data.

Inputs	Expected Output	Actual Outputs	Comments

Implement your program!

Hint:

The GUI for this one can be difficult to follow if you aren't careful – think about a logical and clear layout

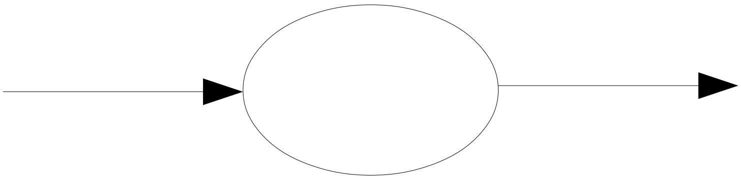
Testing

Use your table above to identify any issues with your program

32. Ker-Ching Change Giver

Design and write a program to simulate the operation of a cash register within a Hotel Shop. It accepts as input the amount paid by the customer as well as the amount due. The program should return as output the change due expressed as coins of the relevant denominations eg.

Amount due = £28.76
Amount tendered £50.00
Change Due = £21.24 as
    1* £20 Note
    0 * £10 Note
    0 * £5 Note
    0 * £2 Coin
    1 * £1 Coin
    0 * 50p
    1 * 20p
    0 * 10p
    0 * 5p
    2 * 2p
    0 * 1p



Pseudocode **Flowchart**

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan
Complete the first 2 columns of the following Test Plan BEFORE implementation. Run the tests to see if your program works.

Inputs	Expected Output	Actual Outputs	Comments

Implement your program!

Hint:

There are brute-force ways to tackle this and there are more elegant methods that produce more compact and efficient code. Which will you choose?

Testing

Use the table above to identify any issues with your program

Explain these terms:

Repeated subtraction...

Integer division...

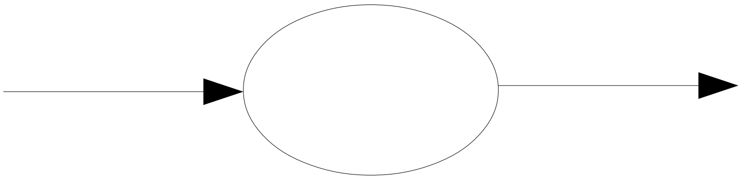
Modulus division...

Array:...

Constant:...

33. Tax Calculator

Write a program to calculate the amount of Tax payable MONTHLY by an employee at a Hotel. An employee can earn £7500 p.a. free of tax after which 10% tax is payable on the next £2500. An employee should then pay 25% tax on the amount up to £38000 p.a. and 40% thereafter. The user can enter his annual salary and the program then shows the monthly pay and tax payable for the employee.



Flowchart

Pseudocode

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. As usual, include valid, invalid and extreme data. For this program, testing the boundaries of each tax level is absolutely vital.

Inputs	Expected Output	Actual Outputs	Comments

Implement your program!

Hint:

You really need to be clear about the maths – work out a few examples first!

Testing

Use the table above to identify any issues with your program

Explain these terms:

Test data..

Valid data...

Invalid data...

Extreme data...

Boundary data...

34. Painter Quoter

You have been asked to develop a program to help give quotations for decorating rooms.

The program will assume that a room is a rectangular box, with the length, width and height being given in metres. The quotation will be simply for the worker's time in decorating and will not include the cost of materials (paints, wallpaper etc).

Walls:

N.B. When working out the costs for decorating the walls, ignore any doors, windows etc – just assume that each wall is a featureless surface for these calculations.

Painting walls costs £1.25 per square metre; wall-papering walls costs £1.75 per square metre. If walls are taller than 2.25 metres then the workers will need to work on ladders and trestles, so the price goes up to £2.00 per square metre for painting and £2.50 per square metre for wall-papering.

In addition:

· painting skirting boards costs 50p per metre length (assume that the skirting board goes all around the base of the room – again ignore gaps for doors etc)

· painting a door costs £5.50

· painting a window costs £10.00

Ceilings:

Painting ceilings costs £2.20 per square metre. If the ceiling is higher than 2.25 metres then the cost rises to £3.00 per square metre.

Floors (optional):

If the customer has wooden floors that they want sanding and varnishing, this costs £4.00 per square metre.

There is an additional charge of £10 for hire of dustsheets to protect furniture and carpets from the decorating. This is a standard charge, irrespective of the size of the room.

Design, write and test a program to fulfil the above specification. You will need to plan the calculations carefully and use sensible test data.

Submit a fully detailed Showcase for your program

35. Thief!

This is a tough one!

A thief has managed to find out the four digits for an online PIN code, but doesn't know the correct sequence needed to hack into the account.

Design and write a program that displays all the possible combinations for any four numerical digits entered by the user. The program should avoid displaying the same combination more than once.

Submit a fully detailed Showcase for your program

36. Treasure Hunt
A treasure hunt game is played on a 5 x 5 grid. One square contains a pot of gold, 5 squares contain deadly creatures (spiders, scorpions etc). At the start of the game, the program randomly hides the treasure and the deadly creatures. The player then chooses squares until he either finds the treasure, or is killed by a deadly creature. The program should show which squares the player has chosen so far and display appropriate messages when he encounters an object or chooses an empty location.

The program gives a "getting warmer" message if the player chooses a location 2 squares from the treasure and a "getting hot" message if he is 1 square away. It gives similar messages for choosing squares close to the deadly creatures

37. Password reset
A password for an online banking system consists of 9 characters, at least one of which must be an upper case letter and at least one of which must be a number. Users must change their password every month. The new password must not contain more than 3 of the same characters used in the current password.

Write a program to compare 2 strings to see if the second is a valid new password. The program should give the following feedback according to how many of the same characters are used in the new password.

4-6:	Invalid password. Your new password contains too many letters from the existing password.
7-8:	Invalid password. Your new password contains almost identical letters to the existing password.
9:	Invalid password. Your new password contains identical letters to the existing password.

38. Fruity
Write a program to simulate a Fruit Machine that displays 3 symbols at random from Cherry, Bell, Lemon, Orange, Star, Skull.

The player starts with £1 credit, with each go costing 20p. If the Fruit Machine "rolls" 2 of the same symbol, the user wins 50p. The player wins £1 for 3 of the same and £5 for 3 Bells. The players loses £1 if 2 skulls are rolled and all of his/her money if 3 skulls are rolled. The player can choose to quit with the winnings after each roll or keep playing until there is no money left!

39. Contagion
Write a program to simulate the spread of a disease in a 20 x 20 area. The program randomly sets one "cell" to be diseased at the start of the game. Each round, every cell is checked and if any cell is next to a diseased cell (including diagonally), it too becomes diseased. The display is updated every round to show diseased/healthy cells.

Every 5 rounds, the player can choose to inject one cell with an antibody. Every round the antibody spreads using the same rules as the disease. Can the player wipe out the disease? How many rounds can the player last before all cells are diseased?

40. Classification
A simple classification system asks a series of Yes/No questions in order to work out what type of animal is being looked at. Eg Does it have 4 legs? Does it eat meat? Does it have stripes? Etc

These systems can often be drawn using a "tree" structure. Carry out some simple research on classification trees, then write a program to help the user decide between the following:

> horse, cow, sheep, pig, dog, cat, lion, tiger, whale, dolphin, seal, penguin, ostrich, sparrow, spider, ant, bee, wasp, termite, octopus, squid

Is there a better way to do this than using 101 IF...ELSE...END IFs?

Develop your classification system for your own area of interest: pop bands; pokemon; cars; footballers; teachers; diseases etc

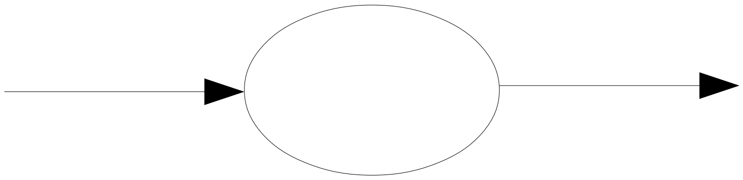
Design Sheet

Use this sheet to plan and design your own programs

Aim of program:

……

……



Flowchart

Pseudocode

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

....

....

....

....

....

....

....

....

....

Variables and Other Data Structures

Name	Datatype	Typical Value	Minimum Value	Maximum Value

Test Plan

Complete the first 2 columns of the following Test Plan BEFORE implementation. As usual, include valid, invalid and extreme data.

Inputs	Expected Output	Actual Outputs	Comments

Implement your program!

Test your program!

Evaluate your program!

Programming Target Tracker

Date	Program	Grade	Target	Evidence of Achieving Target
eg	*FormChooser*	*C*	*Investigate **Select...Case** as an alternative to using multiple nested **Ifs**...*	*Used Select...Case in the Mini-Calc program to choose between operators*

Programming Resources

Background Reading

http://www.i-programmer.info/babbages-bag.html

http://computer.howstuffworks.com/

http://www.howitworks.net/

Tutorials

http://www.i-programmer.info/ebooks/master-visual-basic.html

http://www.java2s.com/Tutorial/VB/0300__2D-Graphics/Catalog0300__2D-Graphics.htm

http://www.dreamincode.net/forums/forum/78-programming-tutorials/

http://www.codeavengers.com/

https://www.futurelearn.com/courses/begin-programming

http://www.w3schools.com/sitemap/default.asp#examples

http://www.w3schools.com/php/default.asp

http://www.w3schools.com/sql/default.asp

http://www.youtube.com/user/Axsied

http://www.tech-recipes.com/rx/category/computer-programming/

Programming Challenges

http://coderbyte.com/CodingArea/Challenges/

https://projecteuler.net/

http://www.olympiad.org.uk/problems.html

http://bebras.org/?q=examples