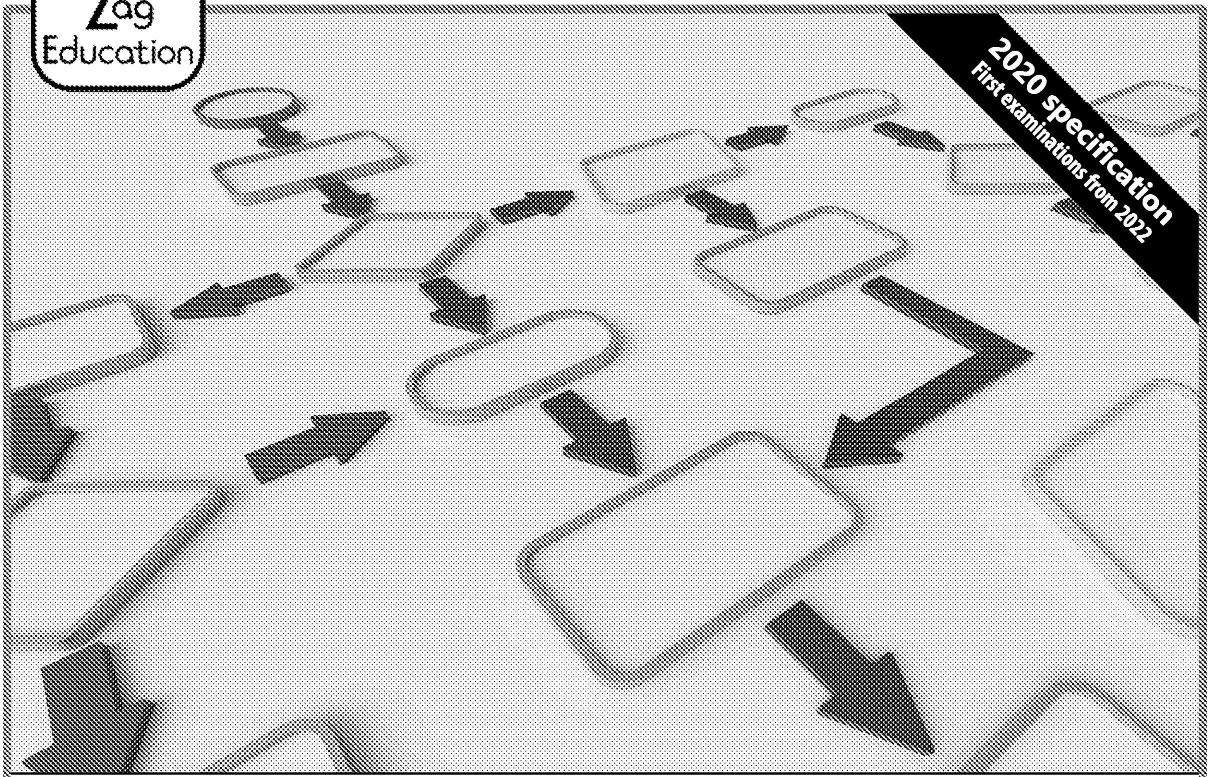
Computer Science GCSE (9–1) | AQA | 8525



Algorithms Resource Pack

for AQA GCSE Computer Science (8525)

Sue Wright

Part 2 – Worksheets & Solutions

zigzageducation.co.uk

POD 10602b

Contents

EXERCISE 1: CHARITY FUNDRAISER – ANALYS

Identify the inputs, process and outputs you would need to know to solve this pr

You have been asked to write a simple algorithm to work out how much money h fundraising activity at school and display the total.

The activities your form took part in were:

- Car washing
- Dog walking

For example, you will know how many cars were washed and what the charge w

INPUTS	PROCESS	

EXERCISE 1: CHARITY FUNDRAISER – ANALYS

Identify the inputs, process and outputs you would need to know to solve this pr

You have been asked to write a simple algorithm to work out how much money h fundraising activity at school and display the total.

The activities your form took part in were:

- Car washing
- Dog walking

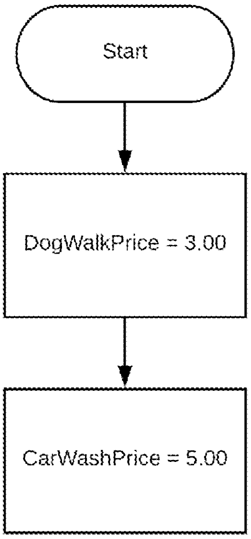
For example, you will know how many cars were washed and what the charge w

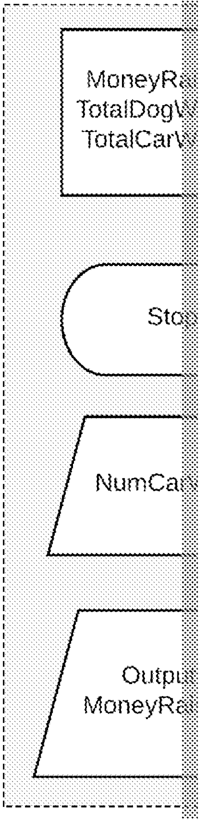
INPUTS	PROCESS	

Exercise 2: Charity Fundraiser – Put the Correct Order

Now that we have identified the inputs, process and outputs needed to solve the chart to give a visual representation of our algorithm. It has been decided that th walking and £5 for car washing.

The flow chart has been started below (on the left); you need to add the remaini correct order.





Exercise 3: Constants or Vari

Complete the table to identify which of the following are constants and which ar
Fill in the last column to explain your answer.

Expression	Constant or variable?	
currentTemp ← 30		
pi ← 3.14159		
diameter ← 34.5		
boilPoint ← 100		
currentShoeSize ← 5.5		
daysInWeek ← 7		
minsInHour ← 60		
playerOneDiceRoll ← 5		
gramToOunce ← 0.0352		
playerName ← "Charlotte"		

EXERCISE 4: HOLIDAY CALCULAT

You have been invited on a four-day holiday to Disneyland Paris with a friend. Th food have been paid for; you need to have money for drinks and souvenirs. You the holiday is a month away so you could have more money by then.

Write an algorithm using **pseudocode** that will calculate how much in euros you You should start by identifying your inputs, process and outputs before attempti

INPUTS	PROCESS

Note: Your answer should show the use of constants, variables, the USERINPUT and PRIN of assigning a value to a variable in pseudocode.

Exercise 5: Holiday Temperature C

You are visiting a member of your family, who lives in Florida, for a holiday in De
temperature will be about 61 ° Fahrenheit; we use Celsius to measure temperatu

Write an algorithm using pseudocode which will allow the user to enter the temp
the equivalent in Celsius to the screen.

*Note: The formula will be (F – 32) * 5/9 = C.*

Identify your inputs, process and outputs first.

Inputs	Process	

Exercise 6: Odds or Evens

Design a simple algorithm that will take in a number from the user and print whe

Hint: a number that is divisible by 2 with no remainder will be even.

Identify your inputs, process and outputs first.

Inputs	Process	

This should be written **BOTH** in pseudocode and as a flow chart.

Flow chart	**Pseudo code**

EXERCISE 7: COLOUR RANGE

Write an algorithm that will take in a number, check that the number is within a
colour. If the number is not in the correct range the algorithm must display an e

- Between 0 to 10 = red
- Between 11 to 20 = green
- Between 21 to 30 = blue

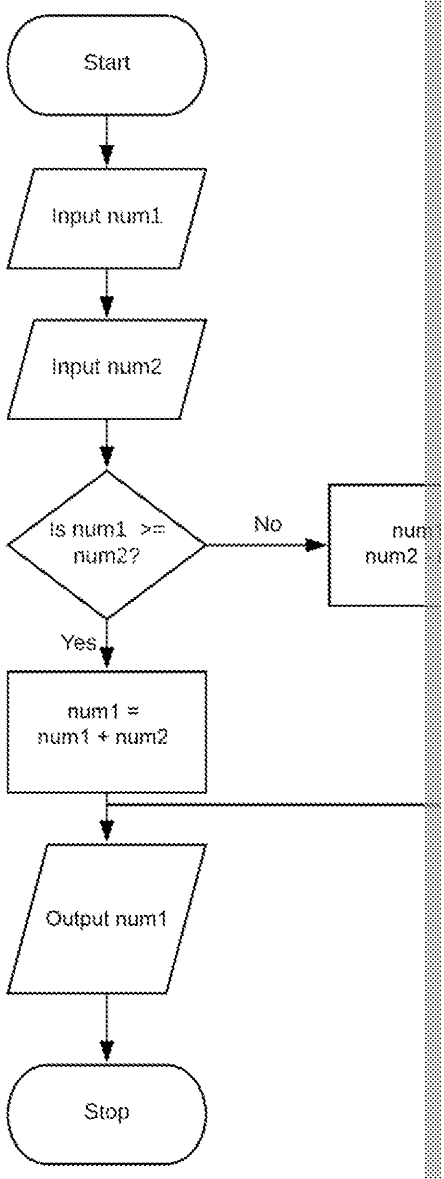
Identify your inputs, process and outputs first, the produce **BOTH** pseudocode a

INPUTS	PROCESS	

Flow chart	Pseudocode

EXERCISE 8: TRACE TABLE 1

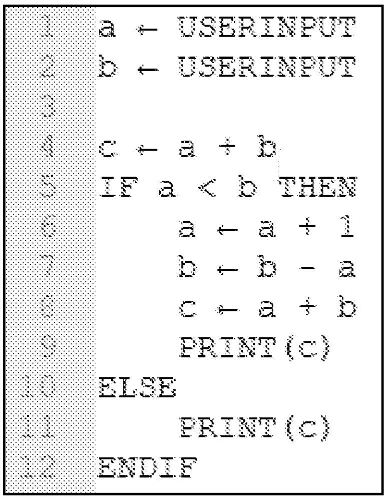
Study the flow chart and complete the trace table below. The first example has b



num1	num2	num1 ≥ num2	nu
5	9	False	4
3	8		
2	10		
12	5		
1	20		
17	3		

EXERCISE 9: TRACE TABLE 2

Read the pseudocode carefully and complete the trace table below.
The first row has been completed for you.

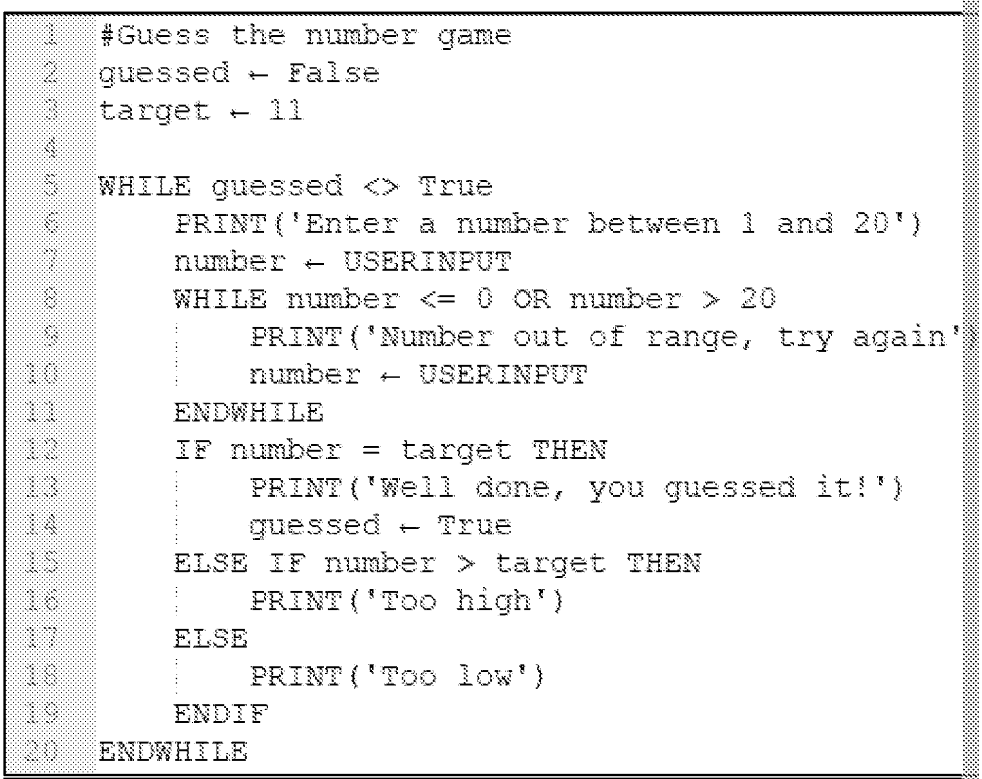


```
a ← USERINPUT
b ← USERINPUT

c ← a + b
IF a < b THEN
    a ← a + 1
    b ← b - a
    c ← a + b
    PRINT(c)
ELSE
    PRINT(c)
ENDIF
```

a	b	c	a < b	a
5	7	12	True	6
15	4			
17	19			
62	49			
23	11			

EXERCISE 10: IDENTIFY THE CONST

Study the example code carefully and complete the table to indicate which lines
of sequence, selection and iteration.



```
1  #Guess the number game
2  guessed ← False
3  target ← 11
4
5  WHILE guessed <> True
6      PRINT('Enter a number between 1 and 20')
7      number ← USERINPUT
8      WHILE number <= 0 OR number > 20
9          PRINT('Number out of range, try again'
10         number ← USERINPUT
11     ENDWHILE
12     IF number = target THEN
13         PRINT('Well done, you guessed it!')
14         guessed ← True
15     ELSE IF number > target THEN
16         PRINT('Too high')
17     ELSE
18         PRINT('Too low')
19     ENDIF
20 ENDWHILE
```

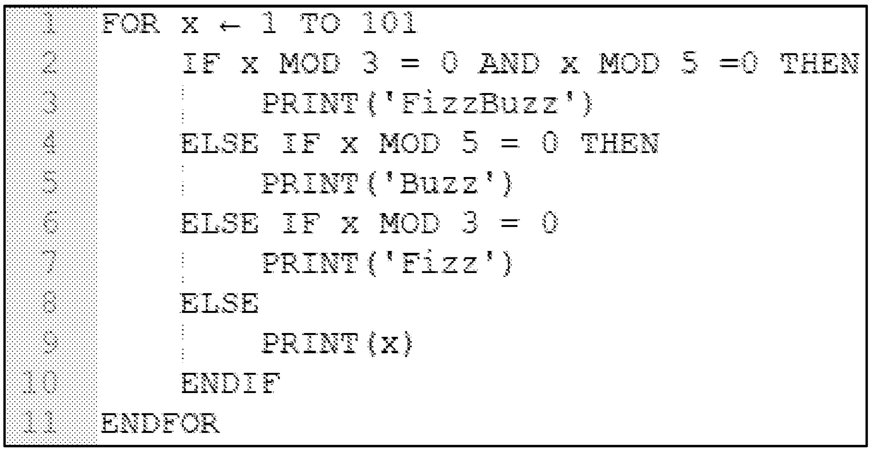
LINE NUMBER(S)	WHICH CONSTRUCT?	EXPLAI
2 and 3	*Sequence*	

EXERCISE 11: FIZZBUZZ TRAC

Complete a trace table for each of the two versions of the FizzBuzz maths game

Explain which version is better, and why.

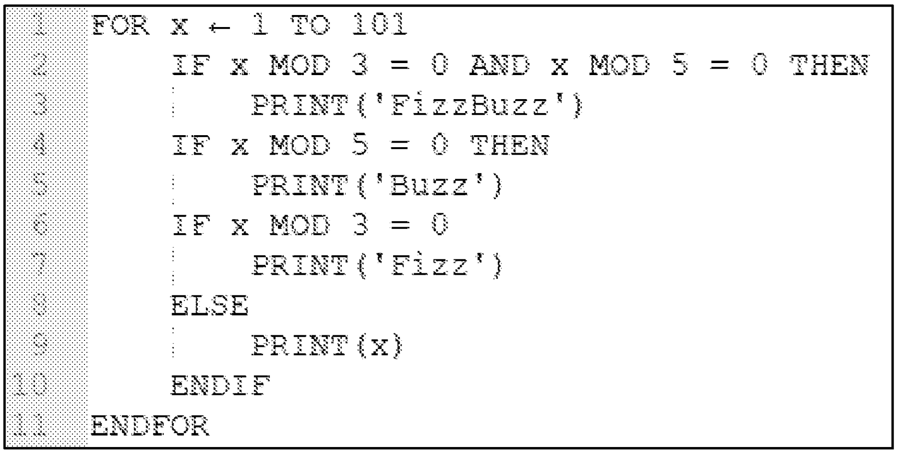
Version 1:



```
FOR x ← 1 TO 101
    IF x MOD 3 = 0 AND x MOD 5 =0 THEN
        PRINT('FizzBuzz')
    ELSE IF x MOD 5 = 0 THEN
        PRINT('Buzz')
    ELSE IF x MOD 3 = 0
        PRINT('Fizz')
    ELSE
        PRINT(x)
    ENDIF
ENDFOR
```

x	X MOD 3 = 0 AND x MOD 5 = 0	X MOD 5 = 0	X MOD
9	*False*	*False*	*Tr*
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

Version 2:



```
FOR x ← 1 TO 101
    IF x MOD 3 = 0 AND x MOD 5 = 0 THEN
        PRINT('FizzBuzz')
    IF x MOD 5 = 0 THEN
        PRINT('Buzz')
    IF x MOD 3 = 0
        PRINT('Fizz')
    ELSE
        PRINT(x)
    ENDIF
ENDFOR
```

x	X MOD 3 = 0 AND x MOD 5 = 0	X MOD 5 = 0	X MOD
9	*False*	*False*	*Tr*
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

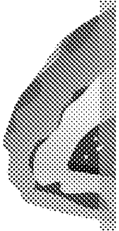
Which version is better and why?

Exercise 12: Dial a Pizza

Dial a Pizza wants a system that is easy to follow to make sure all the right quest
completed and the correct waiting time is given to the customer, based on their

A pizza order is not **complete** until the following questions have been answered:

- Customer address recorded
- Thin, thick or stuffed crust base recorded
- Vegetarian or meat recorded
- Waiting time advised



The times for cooking pizzas are:

- Thin – 10 minutes
- Thick – 15 minutes
- Stuffed crust – 18 minutes

In this exercise you need to create your algorithm using a flow chart (on a separa
correct symbols and arrows.

You will need to think about using 'flag' variables and your answer should use all

Exercise 12: Dial a Pizza

Dial a Pizza wants a system that is easy to follow to make sure all the right quest
completed and the correct waiting time is given to the customer, based on their

A pizza order is not **complete** until the following questions have been answered:

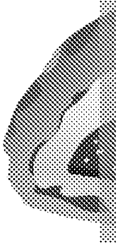
- Customer address recorded
- Thin, thick or stuffed crust base recorded
- Vegetarian or meat recorded
- Waiting time advised

The times for cooking pizzas are:

- Thin – 10 minutes
- Thick – 15 minutes
- Stuffed crust – 18 minutes

In this exercise you need to create your algorithm using a flow chart (on a separa
correct symbols and arrows.

You will need to think about using 'flag' variables and your answer should use all



EXERCISE 13: COUNT UNTIL ZE

Write an algorithm using pseudocode which uses sequence, selection and iterati

The algorithm must continue to ask the user for a number and continue to add t is entered. The total of all the numbers entered (except the 0) must be print tog were entered.

EXERCISE 14: CALCULATE FAR

Midcentral Metrolink has installed a new system for paying fares using a contac loaded with money. The tram fares are calculated as follows:

5 miles or less	£2.00
5–10 miles	£3.25
Above 10 miles	£4.75

When the card is swiped at the start of the journey the tram station identity cod card reader device in the ticket booth. At the end of the journey, the card is swip exit barrier calculates the fare using a data structure called TramMatrix to find th tram stations and deducts the fare from the balance on the smartcard.

Passengers are offered discounts for off-peak travel:

- 10% between 10am and 4pm Monday to Friday
- 15% all day Saturday and Sunday

An example of the TramMatrix is shown here:

STATIONID	DISTANCE (TO NEXT STATION)
MCS001	3.5
MCS002	3
MCS003	2.5
MCS004	4

Note: If the journey starts a StationID MCS002 the total

Study the pseudocode algorithm carefully and answer the following questions o

```
TramStart ← CARD READER

TramMatrix ← [['MCS001', 'MCS002','MCS003','MCS004'],[3.5,3,

TramEnd ← CARD READER
index ← 0

#Card Reader records the index position of the station in th
TramMatrix
#Calculate distance from TramStart to TramEnd


FOR station ← TramStart TO LEN(TramMatrix)
    IF TramStart = station THEN
        Distance ← TramMatrix[1][index]
    ELSE
        index ← index + 1
        Distance ← Distance + TramMatrix[1][index]
    ENDIF
ENDFOR


IF Distance < 5 THEN
    fare ← 2.00
ELSE IF Distance > 10 THEN
    fare ← 4.75
ELSE
    fare ← 3.25
ENDIF

#Calculate discount

PRINT(' Ticket fare is £')
PRINT(fare)
PRINT('Thank you for choosing Midcentral Metrolink')
```

Questions

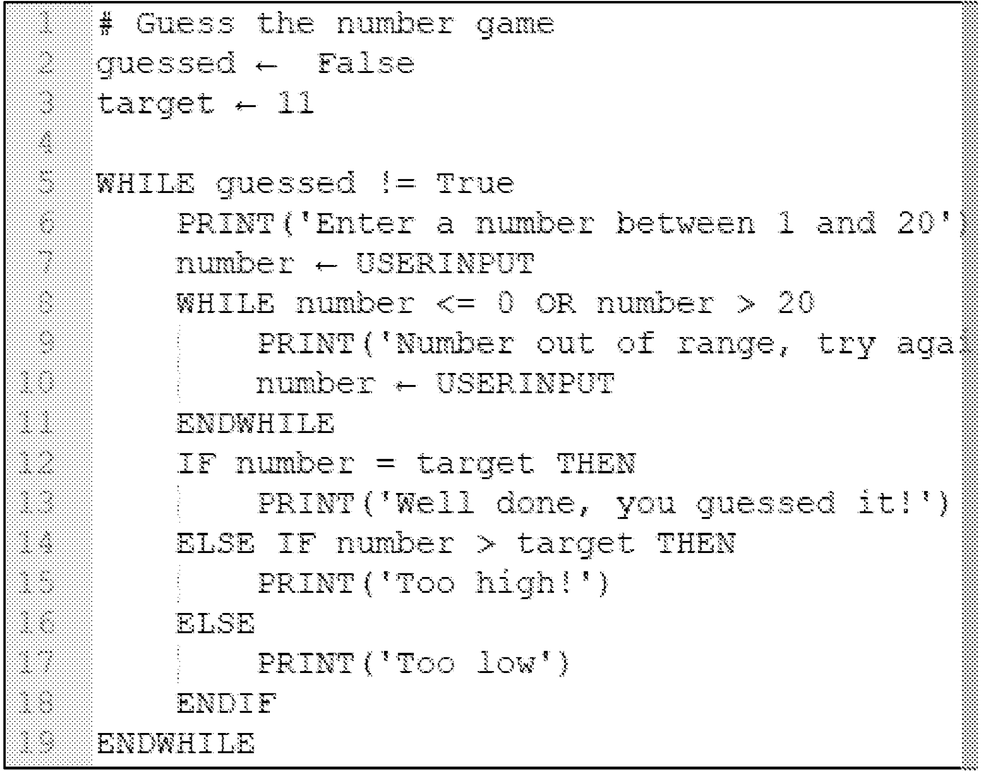
1. The algorithm currently continues adding up the distances instead of sto
   Identify the line where the error occurs and explain how to correct this.

2. The discount functionality has not yet been added. Write the pseudocod
   listed above.

   Hint: The variable name 'Time' may be useful in this answer.

EXERCISE 15: GUESSING GAME USING SU

The code below shows an example of nested iteration as well as demonstrating h iteration can be combined.



```
# Guess the number game
guessed ← False
target ← 11

WHILE guessed != True
    PRINT('Enter a number between 1 and 20')
    number ← USERINPUT
    WHILE number <= 0 OR number > 20
        PRINT('Number out of range, try aga
        number ← USERINPUT
    ENDWHILE
    IF number = target THEN
        PRINT('Well done, you guessed it!')
    ELSE IF number > target THEN
        PRINT('Too high!')
    ELSE
        PRINT('Too low')
    ENDIF
ENDWHILE
```

On a separate piece of paper, re-write this algorithm using subroutines, to:

- allow a user to enter a new target number and return the target
- ask the user for their guess and return the guess

The target and the guess should be used as 'parameters' for the third subroutine prints suitable messages.

Hint: You will need to call all three subroutines at least once.

EXERCISE 16: STRINGS AND SUBSTR

Write the following subroutines using pseudocode:

1. A subroutine which will ask for a string between 10 and 16 characters.
    a. The subroutine must check that a valid string has been entered a
    b. The string entered must be returned from the subroutine.
2. A subroutine that will accept the string (from your first subroutine) as a p starting point and the end point for a substring.
    a. If the start or end point is not valid (because the string is not lon must be shown and the user asked again until a valid start or enc
    b. The original string and the substring should then be printed with

Hint: You will need to check the length of the string in (1) and create a substring (from the
For example, I might enter 'hashtagged' as myString and use SUBSTRING (4, 10, myString

EXERCISE 17: AREA TESTER

You are planning a program that will help younger students test their ability to c
rectangles and triangles.

1. The program must allow a user to choose whether they are testing them
2. The user must enter R to test rectangles, T to test triangles or X to exit.
3. The program must allow the student to enter the length and width for a
   a triangle, and then enter their answer.
4. If the answer is incorrect, they have two more attempts before the corre
5. If the answer entered is correct, they can choose between rectangles or
   program.

Your answer must use subroutines and be presented in a flow chart (on a separa

EXERCISE 17: AREA TESTER

You are planning a program that will help younger students test their ability to c
rectangles and triangles.

1. The program must allow a user to choose whether they are testing them
2. The user must enter R to test rectangles, T to test triangles or X to exit.
3. The program must allow the student to enter the length and width for a
   a triangle, and then enter their answer.
4. If the answer is incorrect, they have two more attempts before the corre
5. If the answer entered is correct, they can choose between rectangles or
   program.

Your answer must use subroutines and be presented in a flow chart (on a separa

EXERCISE 17: AREA TESTER

You are planning a program that will help younger students test their ability to c
rectangles and triangles.

1. The program must allow a user to choose whether they are testing them
2. The user must enter R to test rectangles, T to test triangles or X to exit.
3. The program must allow the student to enter the length and width for a
   a triangle, and then enter their answer.
4. If the answer is incorrect, they have two more attempts before the corre
5. If the answer entered is correct, they can choose between rectangles or
   program.

Your answer must use subroutines and be presented in a flow chart (on a separa

EXERCISE 18: PASSWORD CHECKER VA

In the 'Flow Charts and Subroutines' chapter there is an example of a simple pas

You now need to write a program that will allow the user to enter EITHER an inte must keep count of the number of integers and characters entered to ensure tha more characters in length AND contains three or more numbers 0–9.

Using pseudocode write separate subroutines to allow the user to enter the cha the password and then check the password meets the criteria for length and nu must then ask for the password to be entered again to check whether it matche

Remember to correctly call your subroutines where appropriate.

Hint: Any subroutine can be used more than once in your main program.

EXERCISE 19: ENCRYPTION CIPH

On a separate piece of paper, write an algorithm in pseudocode that will encrypt capital letters only. Your answer must use subroutines. The algorithm should:

1. Ask for the message to be encrypted
2. Ask for a substitute number between 1 and 26
   a. Produce an error message if this number is not in the correct range
   b. Repeat until a suitable number is entered
3. Print the answer as a string, together with the original message

If any characters in the original message are not in capitals, then a question mark encrypted string.

Hint: You will need to use concatenation in this exercise. How will you know a character i

EXERCISE 19: ENCRYPTION CIPH

On a separate piece of paper, write an algorithm in pseudocode that will encrypt capital letters only. Your answer must use subroutines. The algorithm should:

1. Ask for the message to be encrypted
2. Ask for a substitute number between 1 and 26
   a. Produce an error message if this number is not in the correct range
   b. Repeat until a suitable number is entered
3. Print the answer as a string, together with the original message

If any characters in the original message are not in capitals, then a question mark encrypted string.

Hint: You will need to use concatenation in this exercise. How will you know a character i

EXERCISE 19: ENCRYPTION CIPH

On a separate piece of paper, write an algorithm in pseudocode that will encrypt capital letters only. Your answer must use subroutines. The algorithm should:

1. Ask for the message to be encrypted
2. Ask for a substitute number between 1 and 26
   a. Produce an error message if this number is not in the correct range
   b. Repeat until a suitable number is entered
3. Print the answer as a string, together with the original message

If any characters in the original message are not in capitals, then a question mark encrypted string.

Hint: You will need to use concatenation in this exercise. How will you know a character i

EXERCISE 20: SIMPLE BATTLESH

In this exercise you will be creating the algorithm for a simple battleships game usi
arrays to store the position of ships, and a random number generator to choose yo

This should be written in pseudocode (on a separate piece of paper) and use subroutines.

1. Create a 2D array of 5 rows × 5 zeros, e.g.
   `row 1 ← [0, 0, 0, 0, 0].`

2. Create arrays with the locations for your ships.

   a. Cruisers need 4 squares on the grid – you have one cruiser
   b. Submarines need 3 squares – you have two submarines
   c. Destroyers need 2 squares – you have two destroyers

   Example:
   `cruiser ← [[0, 0], [0, 1], [0, 2], [0, 3]]`

3. Your algorithm must randomly calculate which element (row) to look at
   each element.

4. Each time a correct location is found, the algorithm must print a messag

5. The game should run for 10 attempts and then print out how many hits

Hint: Nested loops will be helpful in this exercise.

Exercise 20A: Battleships Extension

Extend the functionality of the simple game so that the same location containing more than once. If the same location is hit again (after the first hit), then the alg not add to the hit count.

Exercise 21: RPG Game Invent

Role-play games are very popular for all ages. They usually involve moving aroun solve puzzles or complete tasks to gain more items to store in an inventory. In or tasks, the player may need to use an item from their inventory.

You need to write an algorithm that will allow players to:

- View the contents of their inventory
- Add items to it
- Use items, i.e. delete them
- Exit from the inventory menu

On a separate piece(s) of paper:

1. Decompose the problem into tasks that can be solved.
2. Write suitable pseudocode subroutines to solve the problem.

✂ -

Exercise 21: RPG Game Invent

Role-play games are very popular for all ages. They usually involve moving aroun solve puzzles or complete tasks to gain more items to store in an inventory. In or tasks, the player may need to use an item from their inventory.

You need to write an algorithm that will allow players to:

- View the contents of their inventory
- Add items to it
- Use items, i.e. delete them
- Exit from the inventory menu

On a separate piece(s) of paper:

1. Decompose the problem into tasks that can be solved.
2. Write suitable pseudocode subroutines to solve the problem.

✂ -

Exercise 21: RPG Game Invent

Role-play games are very popular for all ages. They usually involve moving aroun solve puzzles or complete tasks to gain more items to store in an inventory. In or tasks, the player may need to use an item from their inventory.

You need to write an algorithm that will allow players to:

- View the contents of their inventory
- Add items to it
- Use items, i.e. delete them
- Exit from the inventory menu

On a separate piece(s) of paper:

1. Decompose the problem into tasks that can be solved.
2. Write suitable pseudocode subroutines to solve the problem.

EXERCISE 22: MUSIC GIG

Up-and-coming band ***I Didn't Kno*** helicopter to play their gig at Lord holding a large music festival.

The safest place to land the helico of a lake which is connected to the bridge. They are due to perform a from the island to the stage. The b at one time and, unfortunately, no one torch.

They are due on stage shortly and need to get everyone across to the stage as qu rush and the light is fading they must cross in the minimum time possible and m

The bridge is too long for the torch to be thrown back to the others; it must be c members have different fitness levels, which means they all cross at different sp minute, Bob in 2 minutes, Clair in 5 minutes and Danni in 8 minutes (as she sprai

Explain how you would solve this problem in the shortest possible time.

EXERCISE 23: FILL IN THE B

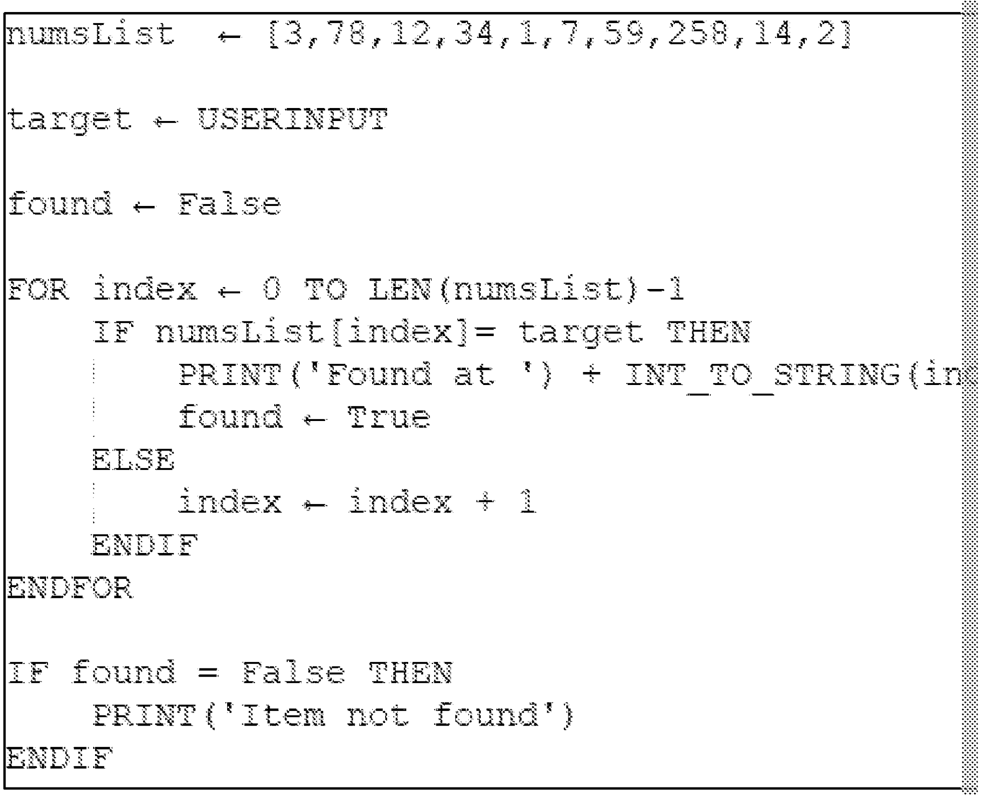
Correct the linear search algorithm below so that it stops when the item has been found.
Complete the blank spaces and check that the algorithm will run correctly when it is called.

```
nameArray ← ['Keiran','Taisha','Emily','Wyatt','Ryan','Zoe','Be
             'Grace','Adam']

target□

PROCEDURE searchList(name,list)
    found ← False
    index□

□ index < □ AND □
        IF list[index]= name THEN
            found ← True
            PRINT('Found')
        ELSE
            index ← index +1
        ENDIF
    END□

    IF found = False THEN
        PRINT('Name not found')
    ENDIF
END PROCEDURE


searchList(target, nameArray)
```

EXERCISE 24: LINEAR SEARCHES AND TR

Complete the trace table exercises for these linear searches:

Linear search 1:



```
numsList  ← [3,78,12,34,1,7,59,258,14,2]

target ← USERINPUT

found ← False

FOR index ← 0 TO LEN(numsList)-1
    IF numsList[index]= target THEN
        PRINT('Found at ') + INT_TO_STRING(in
        found ← True
    ELSE
        index ← index + 1
    ENDIF
ENDFOR

IF found = False THEN
    PRINT('Item not found')
ENDIF
```

index	found	target
0	False	34

Linear search 2:

```
numsList  ← [3,78,12,1,7,59,258,14,2,34]

target ← USERINPUT

found ← False
index ← 0

WHILE index < LEN(numsList)AND NOT found
    IF numsList[index]= target THEN
        PRINT('Found at ')+ INT_TO_STRING(inde
        found ← True
    ELSE
        index ← index + 1
    ENDIF
ENDWHILE

IF found = False THEN
    PRINT('Item not found')
ENDIF
```

index	found	target
0	False	1

Now explain which is most efficient and why, referring to the pseudocode to hel

EXERCISE 25: BUBBLE SORT EXER

1. Complete the bubble sort for this array: [5, 1, 6, 2, 4, 3].

5	1	6	2	4	3

2. Complete this explanation of how to perform a bubble sort.
Hint: Remember that this sorting algorithm uses ITERATION.

1. Compare the first two elements in the array

2. Is the first element bigger than the second element?

3.

4.

5.

6.

EXERCISE 26: PUT THE BUBBLE SORT FLOW C

Complete the flow chart by writing the correct letter in the empty spaces.

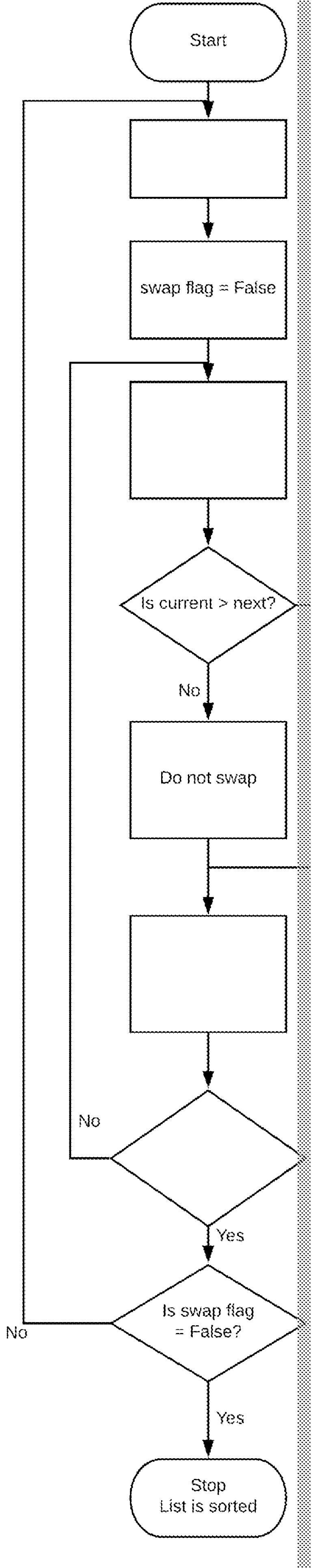
A Move one element along and set this as current element

B Has the last element in array been reached?

C Compare current element with next element

D Look at first element in array

E Swap the two elements



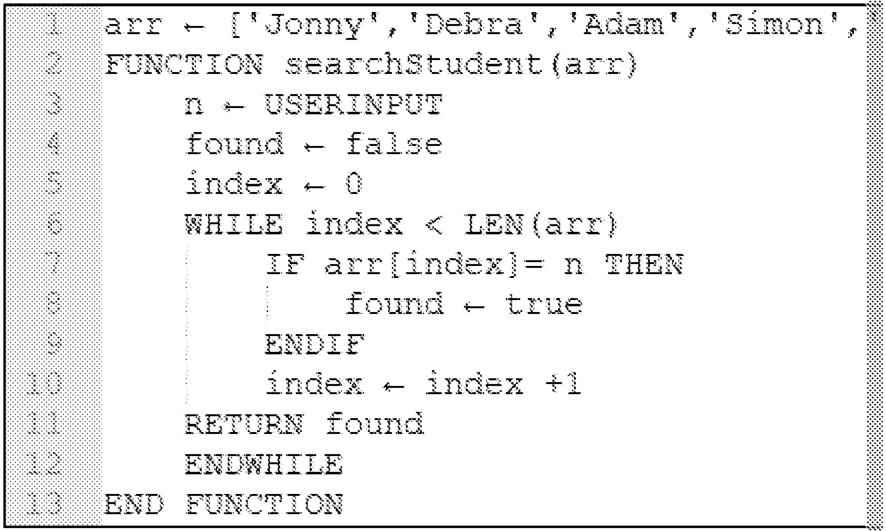
EXERCISE 27: SORTING AND SEARC

1. Complete these data sorts using a merge sort, ensuring that you show al

    a. 67,23,52,6,15,43,11,3

    b. 92,24,2,28,1,7,13,12

2. Samira is writing a simple program to allow a user to enter a name to be pseudocode for the algorithm she wants to use.



```
1  arr ← ['Jonny','Debra','Adam','Simon','
2  FUNCTION searchStudent(arr)
3      n ← USERINPUT
4      found ← false
5      index ← 0
6      WHILE index < LEN(arr)
7          IF arr[index]= n THEN
8              found ← true
9          ENDIF
10         index ← index +1
11     RETURN found
12     ENDWHILE
13 END FUNCTION
```

    a. What type of search is being used?

    b. Describe the algorithm, in terms of its inputs and outputs. What does

    c. The algorithm could be amended to be more efficient. State which lin changed and explain how the change will make the algorithm more e

3. Explain how the bubble sort will work to sort this simple array from:

22	4	13	9	17	1

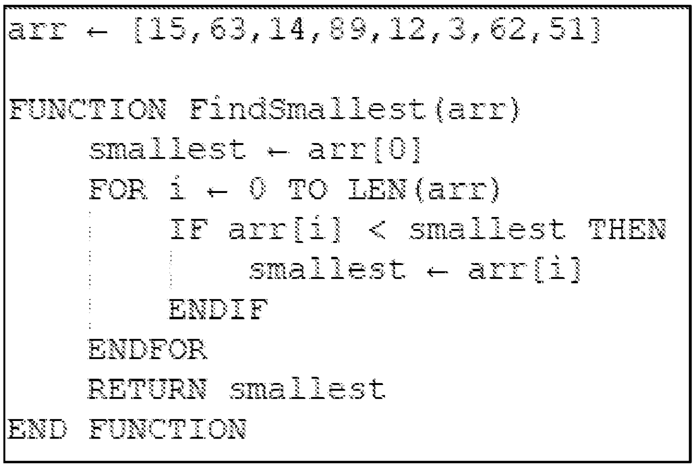
to

1	4	9	13	17	22

The array will start at index position [0].

4. There are two different measurements for the efficiency of an algorithm
Discuss the merge sort and the bubble sort in terms of their time and spa

5. Describe this subroutine in terms of its inputs and outputs. What does it

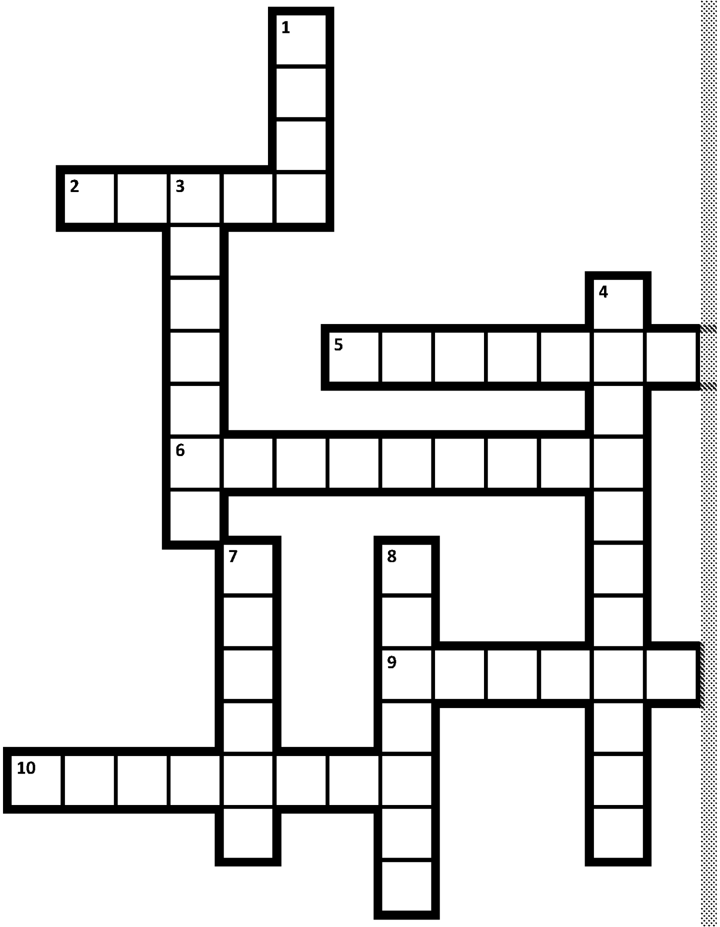


```
arr ← [15,63,14,89,12,3,62,51]

FUNCTION FindSmallest(arr)
    smallest ← arr[0]
    FOR i ← 0 TO LEN(arr)
        IF arr[i] < smallest THEN
            smallest ← arr[i]
        ENDIF
    ENDFOR
    RETURN smallest
END FUNCTION
```

6. Jack has been given homework to write an algorithm to search a variety
Which search method would be most suitable for use with this array, and

[2, 6, 9, 12, 23, 41, 76, 84, 92]

Crossword One



Across

2 Something put into a process (5)

5 An ordered set of steps or instructions (8)

6 A series of instructions that solves a problem in a finite number of steps (9)

9 The result of processing (6)

10 A location in memory where data is stored (8)

Down

1 Something that is

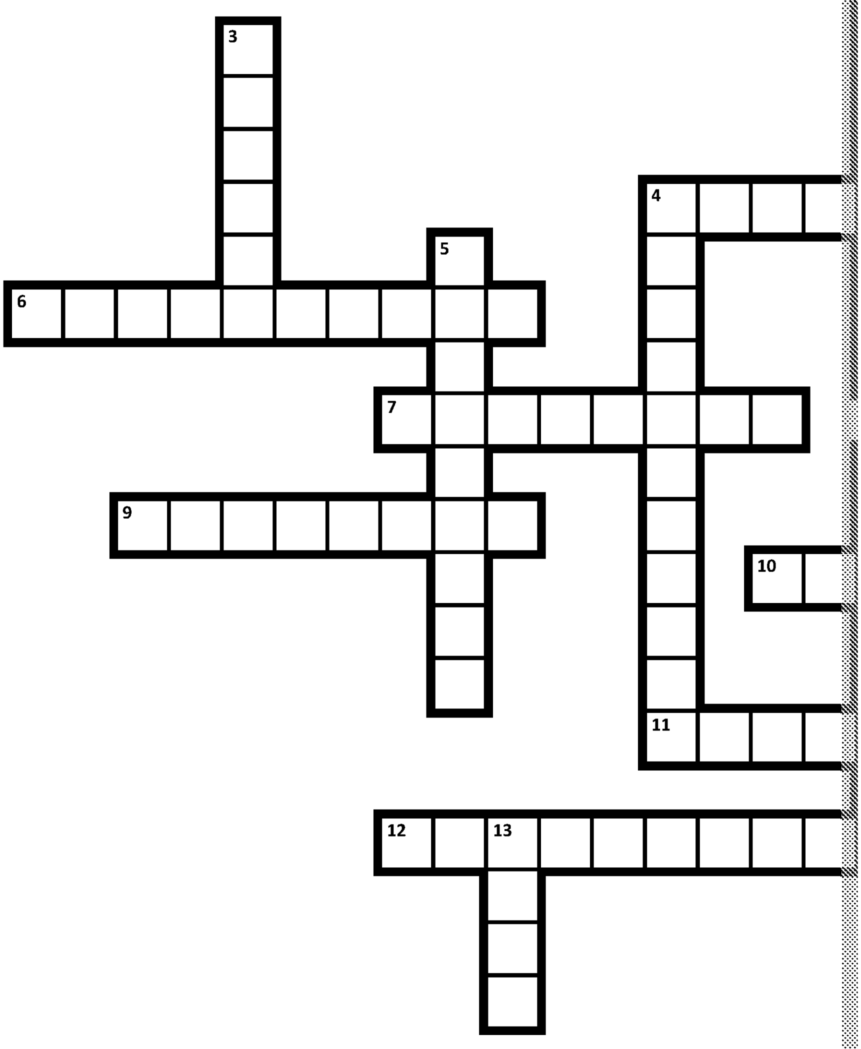
3 Code that tells a co algorithm (7)

4 Written in a way t completely clear (

7 A picture, piece of something (6)

8 A series of steps p (7)

Crossword Two



Across

4 This keyword gets a value into your algorithm from the keyboard (9)

6 This must be unique and meaningful (10)

7 This may change as a program is run (8)

9 The value stored here never changes when a program is run (8)

10 A series of steps performed to achieve a result (7)

11 An ordered set of steps or instructions (8)

12 The term used to describe giving a storage location a value (10)

Down

1 A series of instruct finite number of st

2 Something put into

3 The result of proce

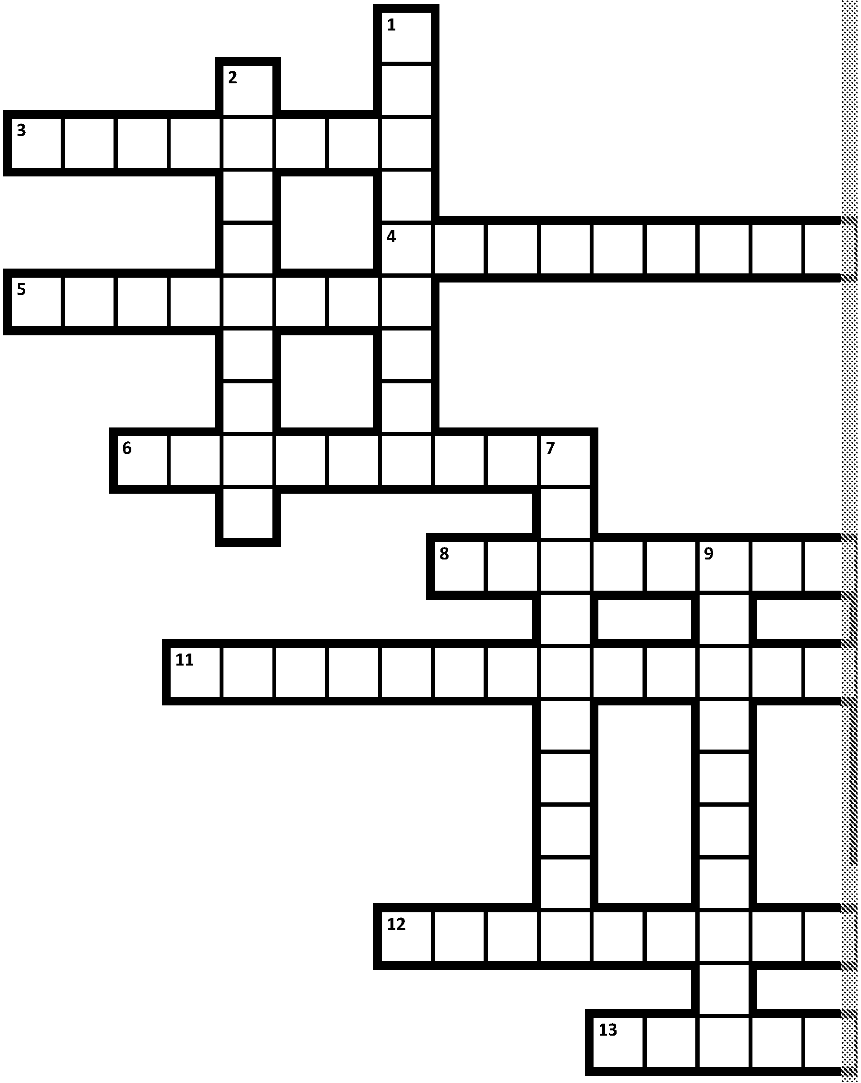
4 Written in a way th what is meant (11)

5 The result of using

8 The result of integ

13 The symbol for mu

CROSSWORD THREE



Across

3 An ordered set of steps or instructions (8)

4 A series of instructions that solves a problem in a finite number of steps that always ends (9)

5 The result of integer division (8)

6 This keyword gets a value into your algorithm from the keyboard (9)

8 Written in a way that makes it completely clear what is meant (11)

11 This means that a Boolean expression which evaluates to True or False runs the loop (8,9)

12 This describes where an algorithm checks whether a condition evaluates to True or False before taking action (9)

13 This may change as a program is run (8)

Down

1 This means instruc program are repea

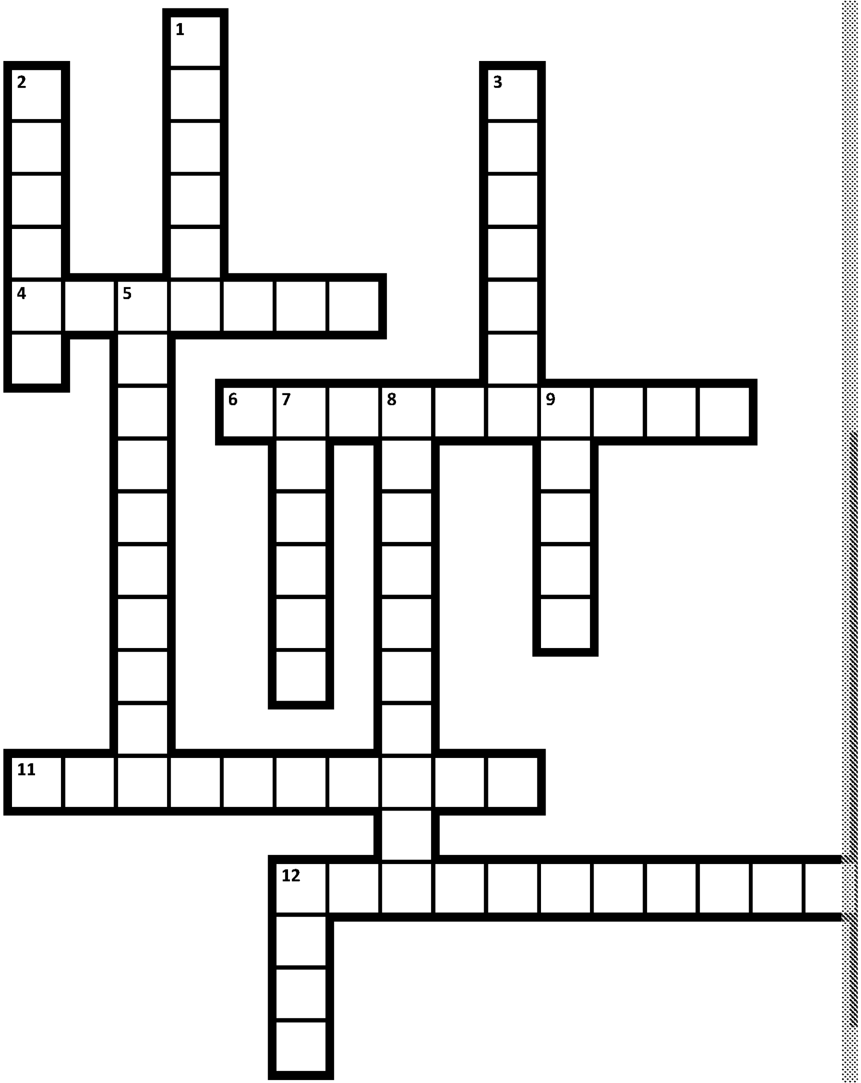
2 The result of using

7 A method to test a no logic errors (5,5

9 This must be uniqu

10 The result of proce

Crossword Four



Across

4 The term used to describe a programming construct, such as a loop, placed inside another programming construct (7)

6 A paper-based method for checking an algorithm (5,5)

11 The name given to a variable (10)

12 The process of joining two strings (13)

Down

1 The result of processing (6)

2 A sequence of characters surr quotation marks (6)

3 Used to describe each item in

5 Used to indicate the start of a (10)

7 This is a feature of a function

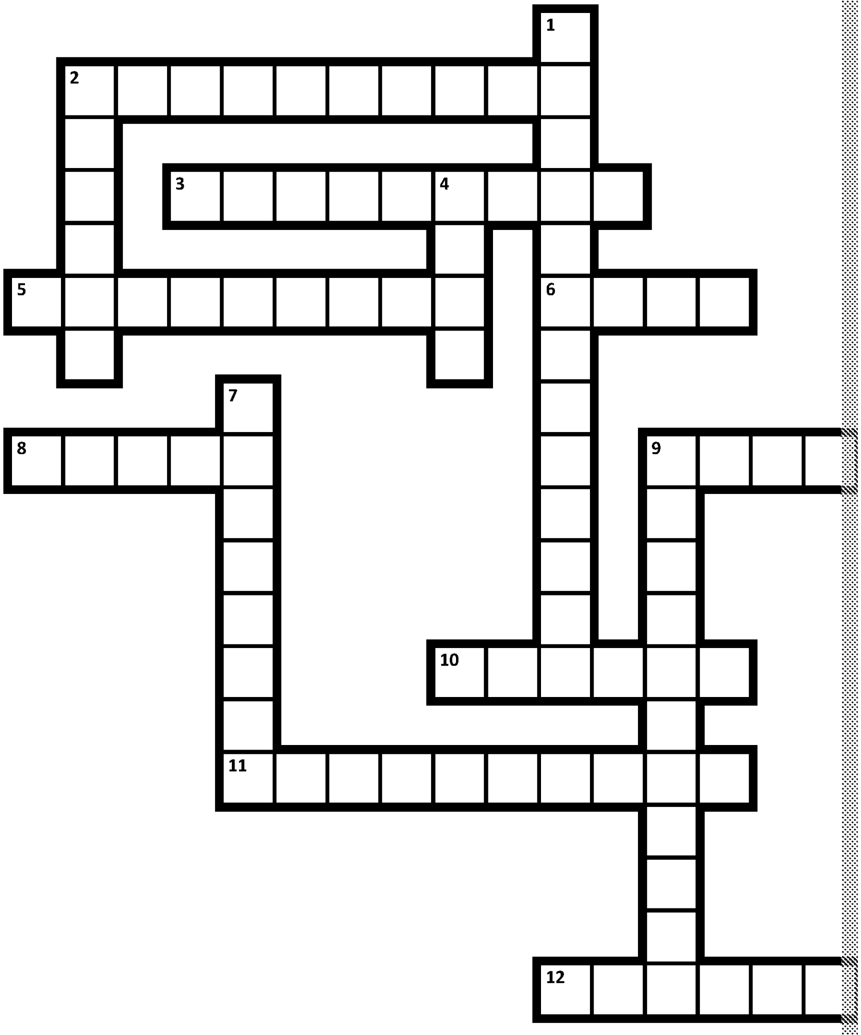
8 The process of changing, for e (10)

9 Data structure to store multip name (5)

10 Written in a way that makes i meant (11)

12 This is the term used to start a program (4)

Crossword Five



Across

2 A problem-solving approach (5,5)

3 The term used to describe repeating a process in an algorithm (9)

5 A series of instructions that solves a problem in a finite number of steps that always ends (9)

6 The process of an algorithm working through a data structure (4)

8 This algorithm has a consistent use of time as the amount of data increases (5)

9 A data structure that can contain many items under one variable name (5)

10 A search method that will only work if the data is sorted (6)

11 The term used to describe how well an algorithm works (10)

12 This algorithm looks at data items in sequence (6)

Down

1 The pro down i

2 This so efficien

4 A meas algorith

7 This me perform set of d

9 The ter all unne

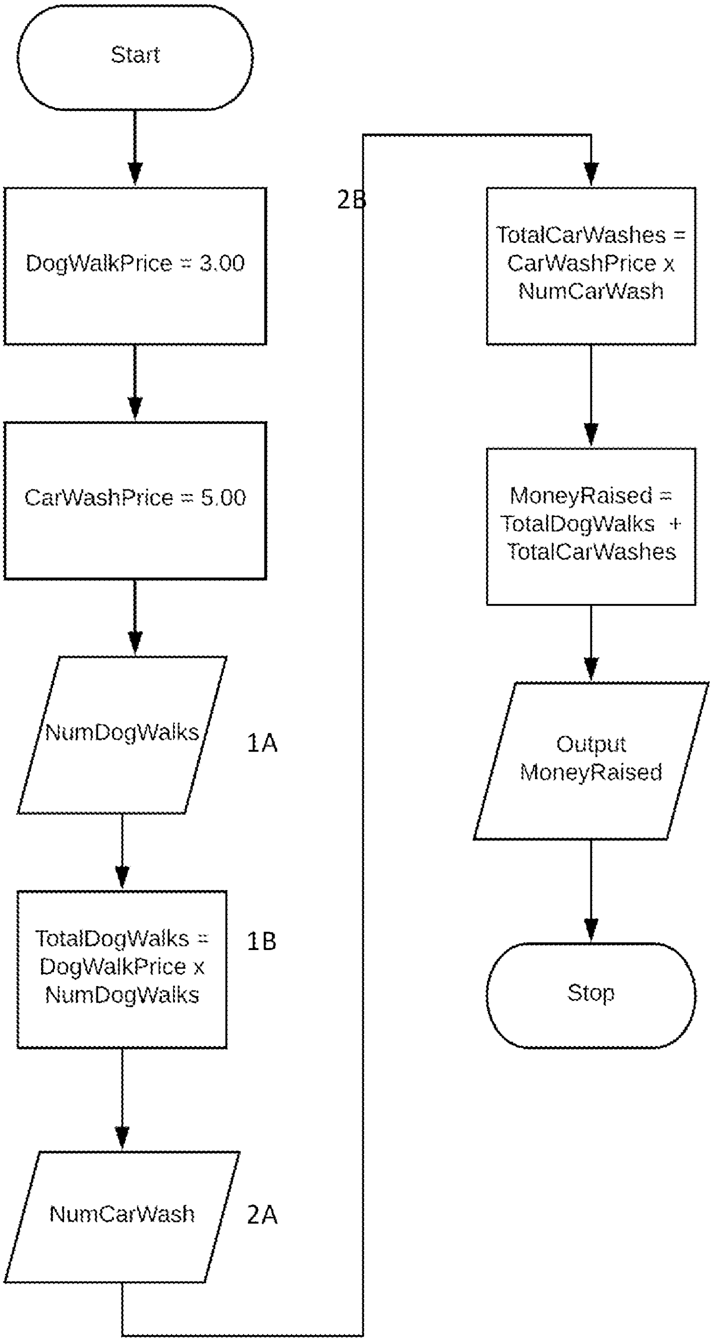
Suggested Answers

Exercises

Exercise 1

Inputs	Process
1. Number of dog walks *2. Number of car washes* *3. Price per dog walk* *4. Price per car wash*	*Total dog walks = No. of dog walks × Price per dog* *Total car washes = No. of car washes × Price per ca* *Money raised = Total dog walks + Total car washes*

Exercise 2



Start
DogWalkPrice = 3.00
CarWashPrice = 5.00
NumDogWalks
1A
TotalDogWalks = DogWalkPrice x NumDogWalks
1B
NumCarWash
2A
2B
TotalCarWashes = CarWashPrice x NumCarWash
MoneyRaised = TotalDogWalks + TotalCarWashes
Output MoneyRaised
Stop

Shapes 1A a
interchange
and still pro
long as 1A c
comes befo

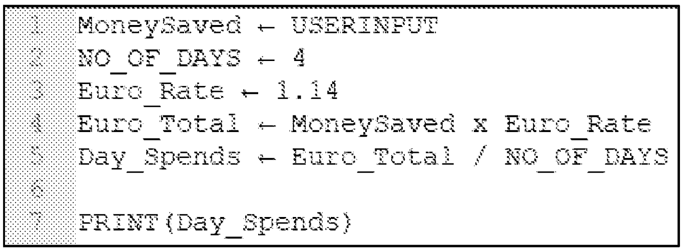
Exercise 3

Expression	Constant or Variable?	R
currentTemp ← 30	*Variable*	*The **identifier** says that this is th this could change when the algo*
pi ← 3.14159	*Constant*	*The mathematical value of pi is*
diameter ← 34.5	*Variable*	*The **identifier** gives a value for a algorithm runs.*
boilPoint ← 100	*Constant*	*The boiling point of water, at se*
currentShoeSize ← 5.5	*Variable*	*The **identifier** gives a value for a change as the algorithm runs.*
daysInWeek ← 7	*Constant*	*The number of days in a week is*
minsInHour ← 60	*Constant*	*The number of minutes in an ho*
playerOneDiceRoll ← 5	*Variable*	*The **identifier** gives a value for a algorithm runs.*
gramToOunce ← 0.0352	*Constant*	*The number of grams to ounces*
playerName ← "Charlotte"	*Variable*	*The **identifier** gives a value for a the algorithm runs.*

Exercise 4

INPUTS	PROCESS	OUTPUTS

MoneySaved
No_of_Days
Euro_rate
	Euro_Total = MoneySaved × Euro_rate
	Day_Spends = Euro_Total / No_of_Days
		Day_Spends



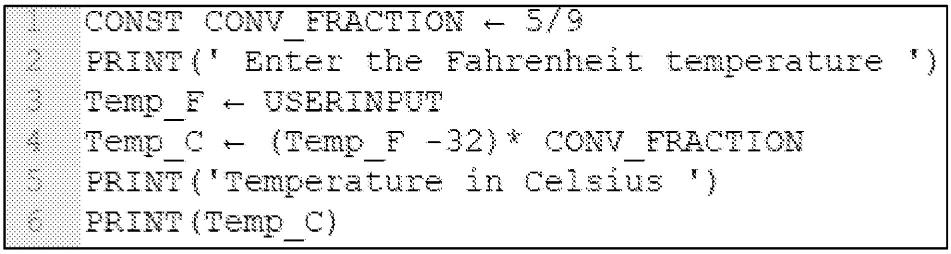
```
MoneySaved ← USERINPUT
NO_OF_DAYS ← 4
Euro_Rate ← 1.14
Euro_Total ← MoneySaved x Euro_Rate
Day_Spends ← Euro_Total / NO_OF_DAYS

PRINT(Day_Spends)
```

Note: Any suitable vari convention are accepta to show the inputs, pro algorithm more clearly added as an input rath

The only CONSTANT w holiday; everything els shown as a variable.

Exercise 5

INPUTS	PROCESS	OUTPUTS
Temp_F Fraction	Temp_C = (Temp_F-32)*Fraction	Temp_C



```
CONST CONV_FRACTION ← 5/9
PRINT(' Enter the Fahrenheit temperature ')
Temp_F ← USERINPUT
Temp_C ← (Temp_F -32)* CONV_FRACTION
PRINT('Temperature in Celsius ')
PRINT(Temp_C)
```

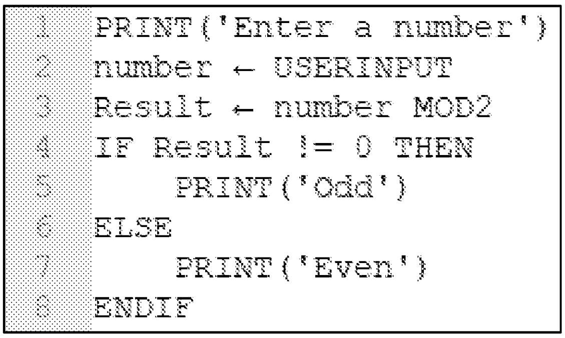
Note: The value of 32 could also be programmed as a constant in this example. The use o necessary in this example but it is good practice for any value that does not change.

Exercise 6

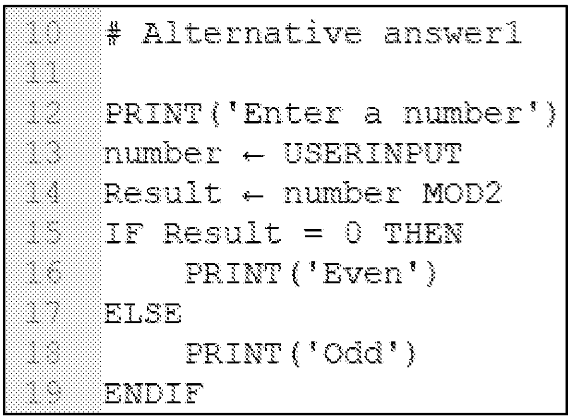
INPUTS	PROCESS	OUTPUTS
number	Result = number MOD 2 If Result <> 0 THEN Output Odd Else Output Even #Alternative Process 1 If Result = 0 THEN Output Even Else Output Odd #Alternative Process 2 If Result >0 THEN Output Odd Else Output Even	Odd or even

This should be written BOTH in pseudocode AND as a flow chart.

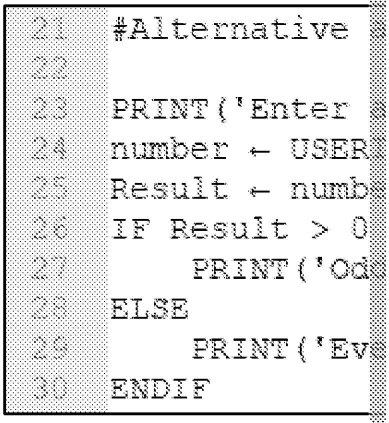
Pseudocode



```
PRINT('Enter a number')
number ← USERINPUT
Result ← number MOD2
IF Result != 0 THEN
    PRINT('Odd')
ELSE
    PRINT('Even')
ENDIF
```

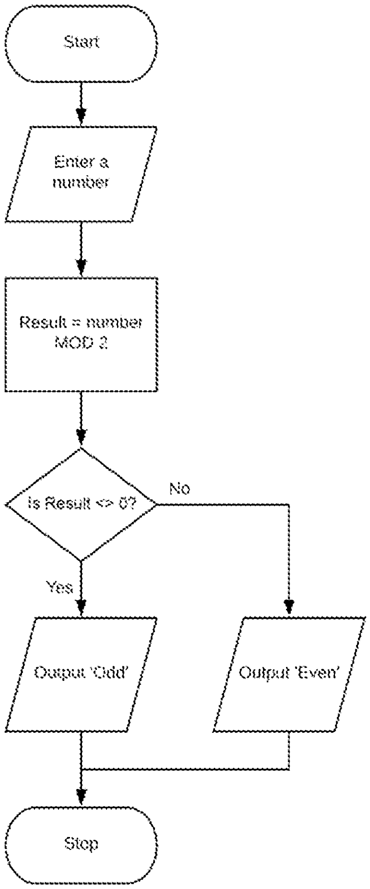


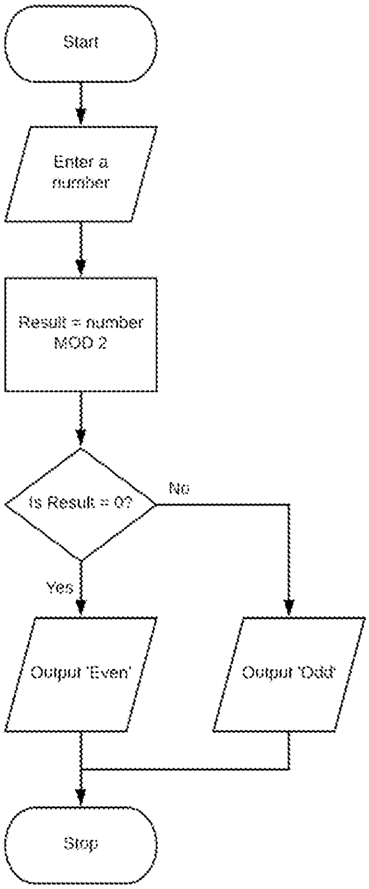
```
# Alternative answer1

PRINT('Enter a number')
number ← USERINPUT
Result ← number MOD2
IF Result = 0 THEN
    PRINT('Even')
ELSE
    PRINT('Odd')
ENDIF
```



```
#Alternative

PRINT('Enter
number ← USER
Result ← numb
IF Result > 0
    PRINT('Od
ELSE
    PRINT('Ev
ENDIF
```

Flow charts



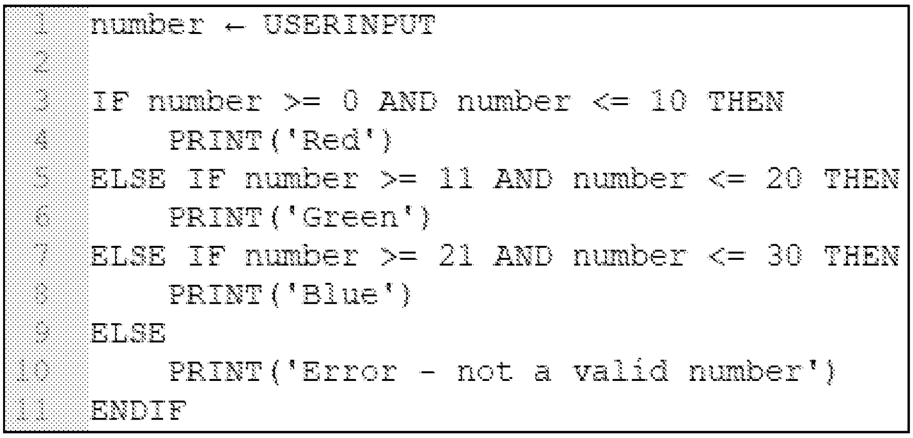


Exercise 7

INPUTS	PROCESS	OUTPUTS
number	If number is between 0 and 10 then output red If number is between 11 and 20 then output green If number is between 21 and 30 then output blue	Red, green or blue Error – not a valid n

This should be written BOTH in pseudocode AND as a flow chart.

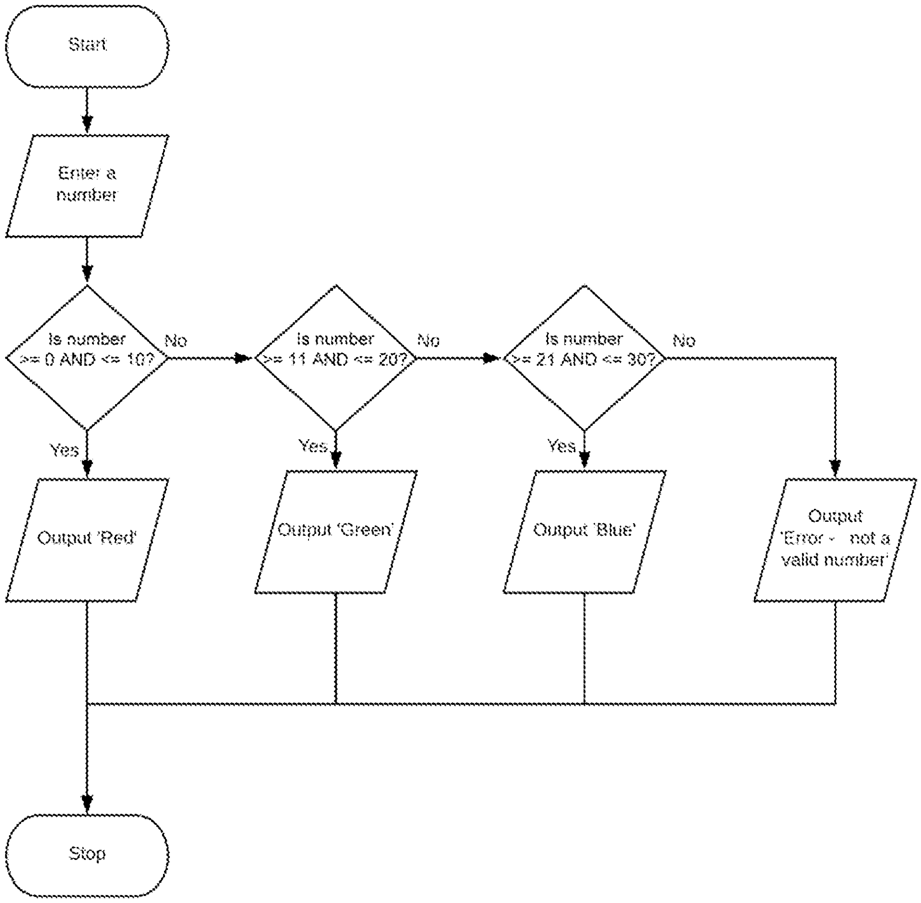
Pseudocode



```
number ← USERINPUT

IF number >= 0 AND number <= 10 THEN
    PRINT('Red')
ELSE IF number >= 11 AND number <= 20 THEN
    PRINT('Green')
ELSE IF number >= 21 AND number <= 30 THEN
    PRINT('Blue')
ELSE
    PRINT('Error - not a valid number')
ENDIF
```

No
th
sy
en
te

Flow chart


Start
Enter a number
Is number >= 0 AND <= 10?
No
Is number >= 11 AND <= 20?
No
Is number >= 21 AND <= 30?
No
Yes
Yes
Yes
Output 'Red'
Output 'Green'
Output 'Blue'
Output 'Error - not a valid number'
Stop

Exercise 8

num1	num2	num1 >= num2	num1	Output
5	*9*	*False*	*4*	*4*
3	*8*	False	5	5
2	*10*	False	8	8
12	*5*	True	17	17
1	*20*	False	19	19
17	*3*	True	20	20

Exercise 9

a	b	c	a < b	a	b	Output
5	*7*	*12*	*True*	*6*	*1*	*7*
15	*4*	*19*	*False*			*19*
17	*19*	36	True	18	1	19
62	*49*	111	False			111
23	*11*	34	False			34

Note: Where values do not change (a,b) they do not need to be repeated in the trace tabl

Exercise 10

Line no.	Construct	Explanation
2 and 3	*Sequence*	The instructions follow one another in sequence.
5 to 20	Iteration	Line 5 shows a WHILE loop using condition-controlled iteration. I
6 and 7	Sequence	The instructions follow one another in sequence.
8 to 11	Iteration	This shows another WHILE loop 'nested' inside the main WHILE loo indefinite iteration as it only stops when the number entered is be
12 to 18	Selection	This is an ELSE-IF statement with three possible options. It controls Lines 5 and 20. When the number entered equals the target, the B True and the condition for the main WHILE loop no longer evaluat

Note: 'Nesting' means combining code together. In this example, an inner WHILE loop is outer WHILE loop between Lines 5 and 20.

Exercise 11

Version 1

X	X MOD 3 = 0 AND x MOD 5 = 0	X MOD 5 = 0	X MOD 3 = 0	OUTPUT
9	*False*	*False*	*True*	*Fizz*
10	False	True	False	Buzz
11	False	False	False	11
12	False	False	True	Fizz
13	False	False	False	13
14	False	False	False	14
15	True	True	True	FizzBuzz
16	False	False	False	16
17	False	False	False	17
18	False	False	True	Fizz
19	False	False	False	19
20	False	True	False	Buzz

Version 2

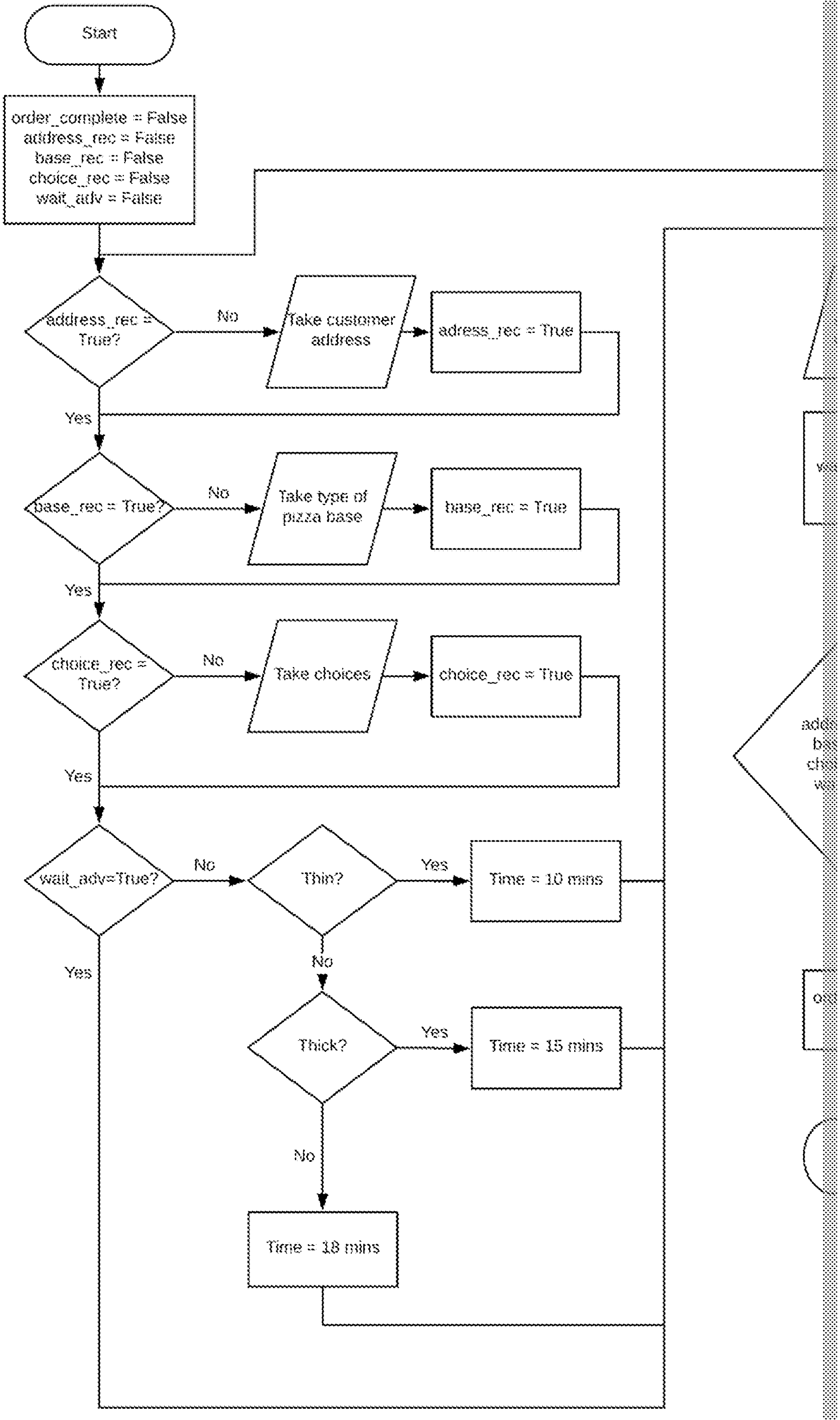
x	X MOD 3 = 0 AND x MOD 5 = 0	X MOD 5 = 0	X MOD 3 = 0	OUTPUT
9	*False*	*False*	*True*	*Fizz*
10	False	True	False	Buzz 10
11	False	False	False	11
12	False	False	True	Fizz
13	False	False	False	13
14	False	False	False	14
15	True	True	True	FizzBuzz Buzz Fizz
16	False	False	False	16
17	False	False	False	17
18	False	False	True	Fizz
19	False	False	False	19
20	False	True	False	Buzz 20

Explain wh

Version 2 d
see in the tr
20 – the IF s
instead of t
one of the t
condition is
these multi

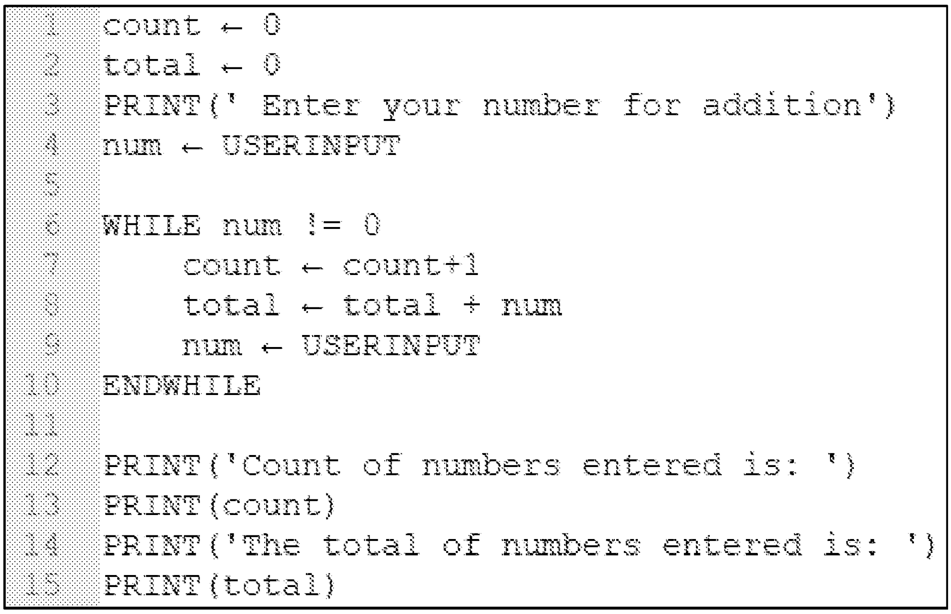
Version 1 is
reasons.

Exercise 12



There are five ‘flags’ set at the start of the process; each of the conditions is checked and the appropriate flag to True. There is a final check at the end of the algorithm; if all four the order is complete and the process finishes.

Exercise 13

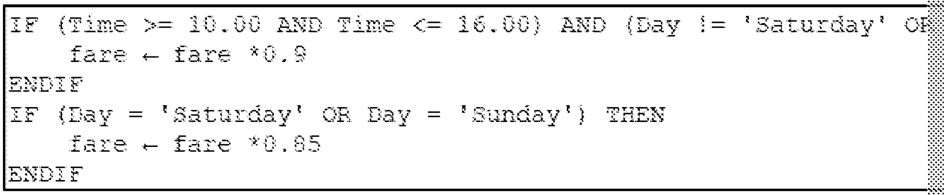


```
count ← 0
total ← 0
PRINT(' Enter your number for addition')
num ← USERINPUT

WHILE num != 0
    count ← count+1
    total ← total + num
    num ← USERINPUT
ENDWHILE

PRINT('Count of numbers entered is: ')
PRINT(count)
PRINT('The total of numbers entered is: ')
PRINT(total)
```

Exercise 14

1. The error is on Line 12 as the FOR loop runs to the end of the TramMatrix. This line s
   `TramStart TO TramEnd`

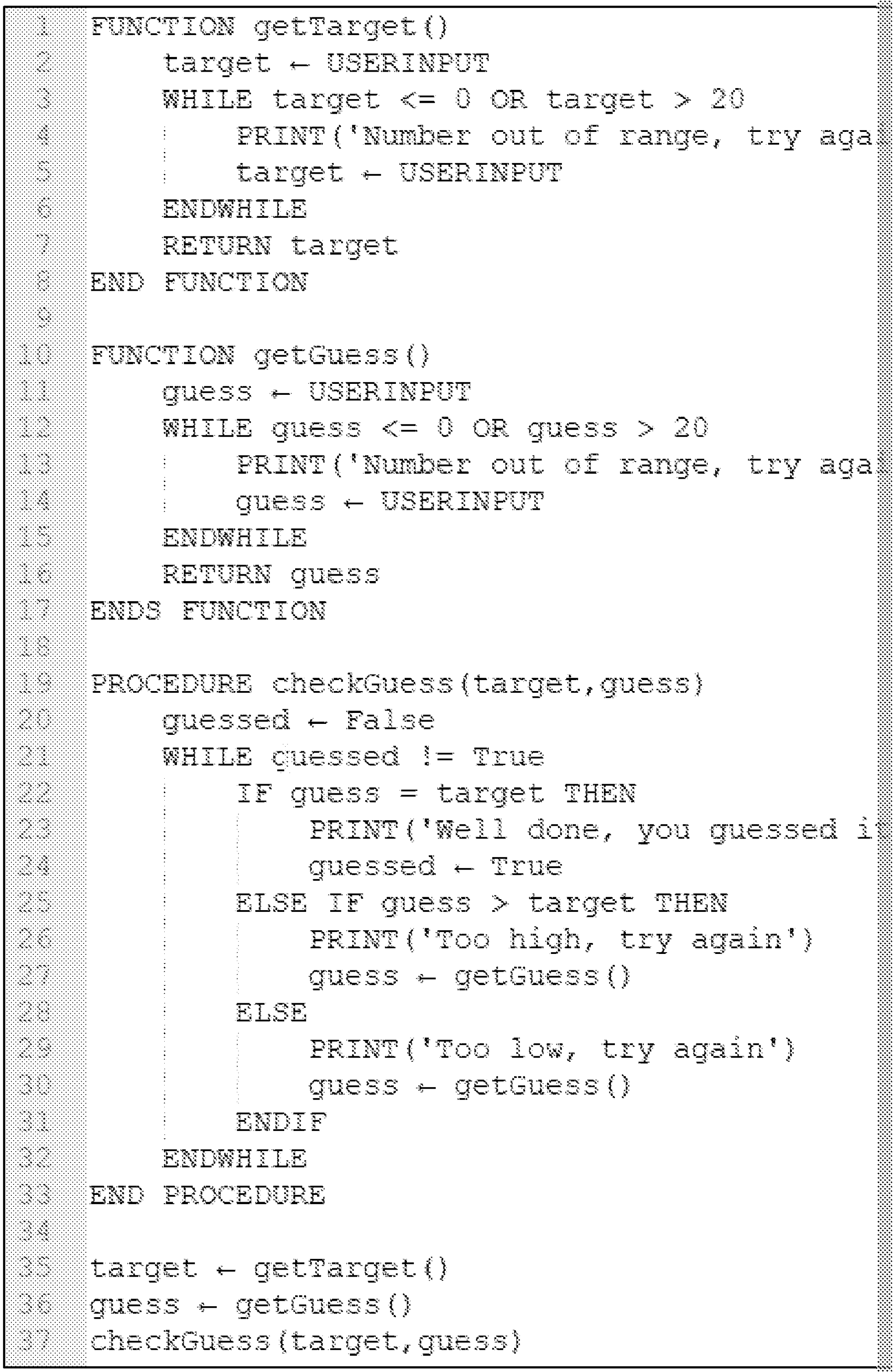
2.



```
IF (Time >= 10.00 AND Time <= 16.00) AND (Day != 'Saturday' OR
    fare ← fare *0.9
ENDIF
IF (Day = 'Saturday' OR Day = 'Sunday') THEN
    fare ← fare *0.85
ENDIF
```

Note: This could also be written using an IF/ELSEIF statement to combine the two IF

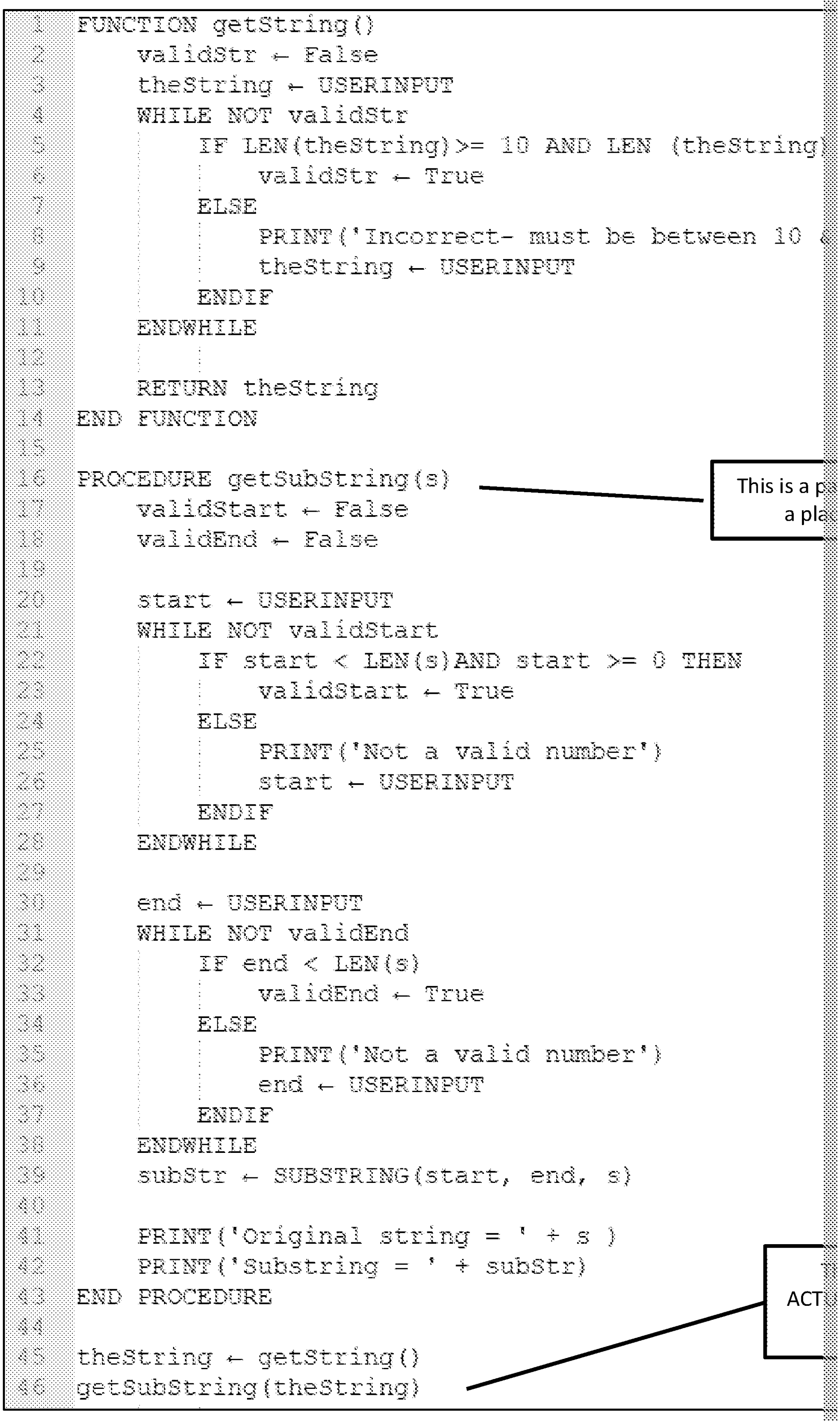
Exercise 15



```
FUNCTION getTarget()
    target ← USERINPUT
    WHILE target <= 0 OR target > 20
        PRINT('Number out of range, try aga
        target ← USERINPUT
    ENDWHILE
    RETURN target
END FUNCTION

FUNCTION getGuess()
    guess ← USERINPUT
    WHILE guess <= 0 OR guess > 20
        PRINT('Number out of range, try aga
        guess ← USERINPUT
    ENDWHILE
    RETURN guess
ENDS FUNCTION

PROCEDURE checkGuess(target,guess)
    guessed ← False
    WHILE guessed != True
        IF guess = target THEN
            PRINT('Well done, you guessed i
            guessed ← True
        ELSE IF guess > target THEN
            PRINT('Too high, try again')
            guess ← getGuess()
        ELSE
            PRINT('Too low, try again')
            guess ← getGuess()
        ENDIF
    ENDWHILE
END PROCEDURE

target ← getTarget()
guess ← getGuess()
checkGuess(target,guess)
```

Exercise 16

Note: The subroutine uses a PARAMETER in the design and uses an ARGUMENT (the actu
subroutine is called.

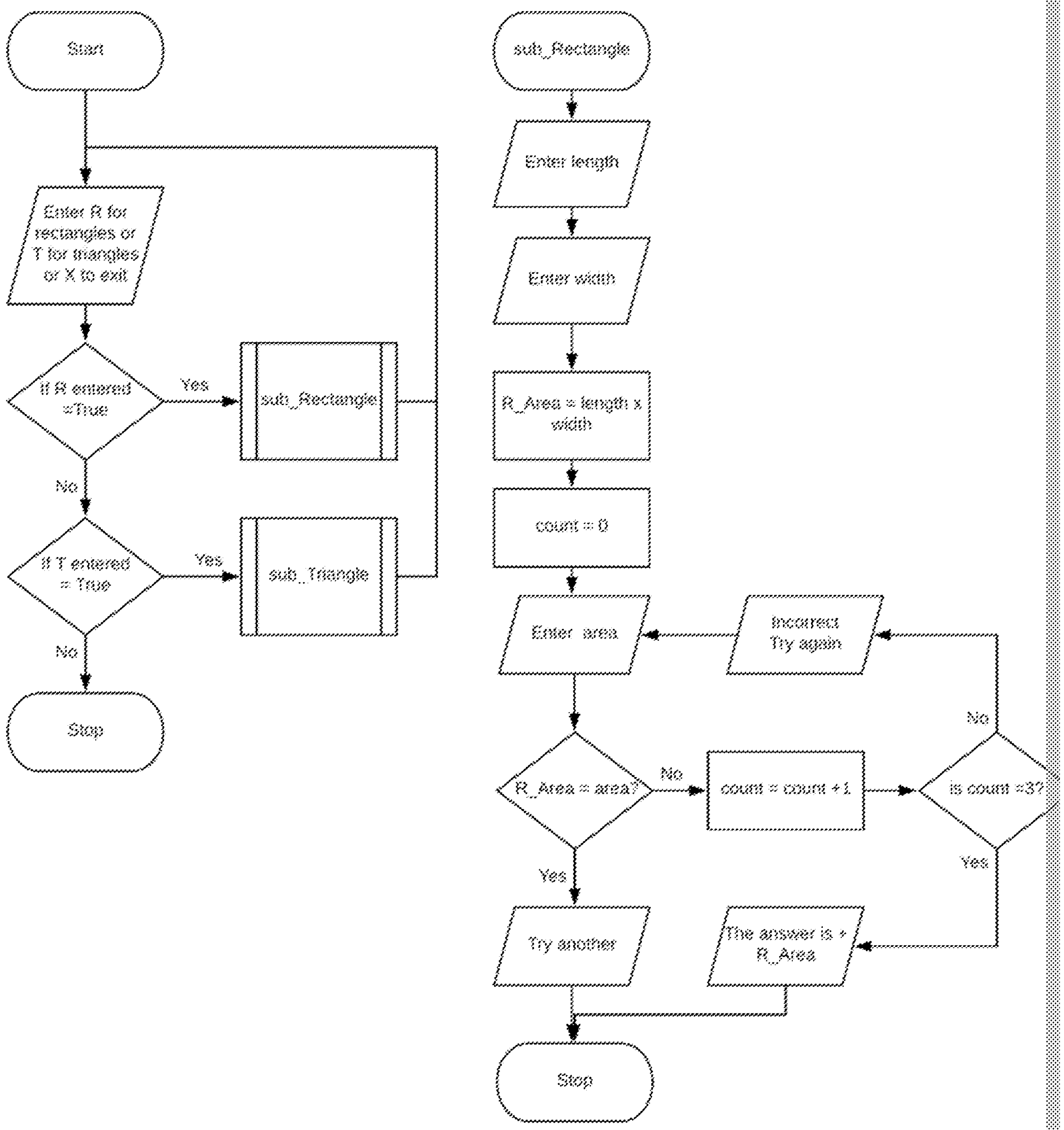


```
FUNCTION getString()
    validStr ← False
    theString ← USERINPUT
    WHILE NOT validStr
        IF LEN(theString)>= 10 AND LEN (theString)
            validStr ← True
        ELSE
            PRINT('Incorrect- must be between 10 &
            theString ← USERINPUT
        ENDIF
    ENDWHILE

    RETURN theString
END FUNCTION

PROCEDURE getSubString(s)
    validStart ← False
    validEnd ← False

    start ← USERINPUT
    WHILE NOT validStart
        IF start < LEN(s)AND start >= 0 THEN
            validStart ← True
        ELSE
            PRINT('Not a valid number')
            start ← USERINPUT
        ENDIF
    ENDWHILE

    end ← USERINPUT
    WHILE NOT validEnd
        IF end < LEN(s)
            validEnd ← True
        ELSE
            PRINT('Not a valid number')
            end ← USERINPUT
        ENDIF
    ENDWHILE
    subStr ← SUBSTRING(start, end, s)

    PRINT('Original string = ' + s )
    PRINT('Substring = ' + subStr)
END PROCEDURE

theString ← getString()
getSubString(theString)
```

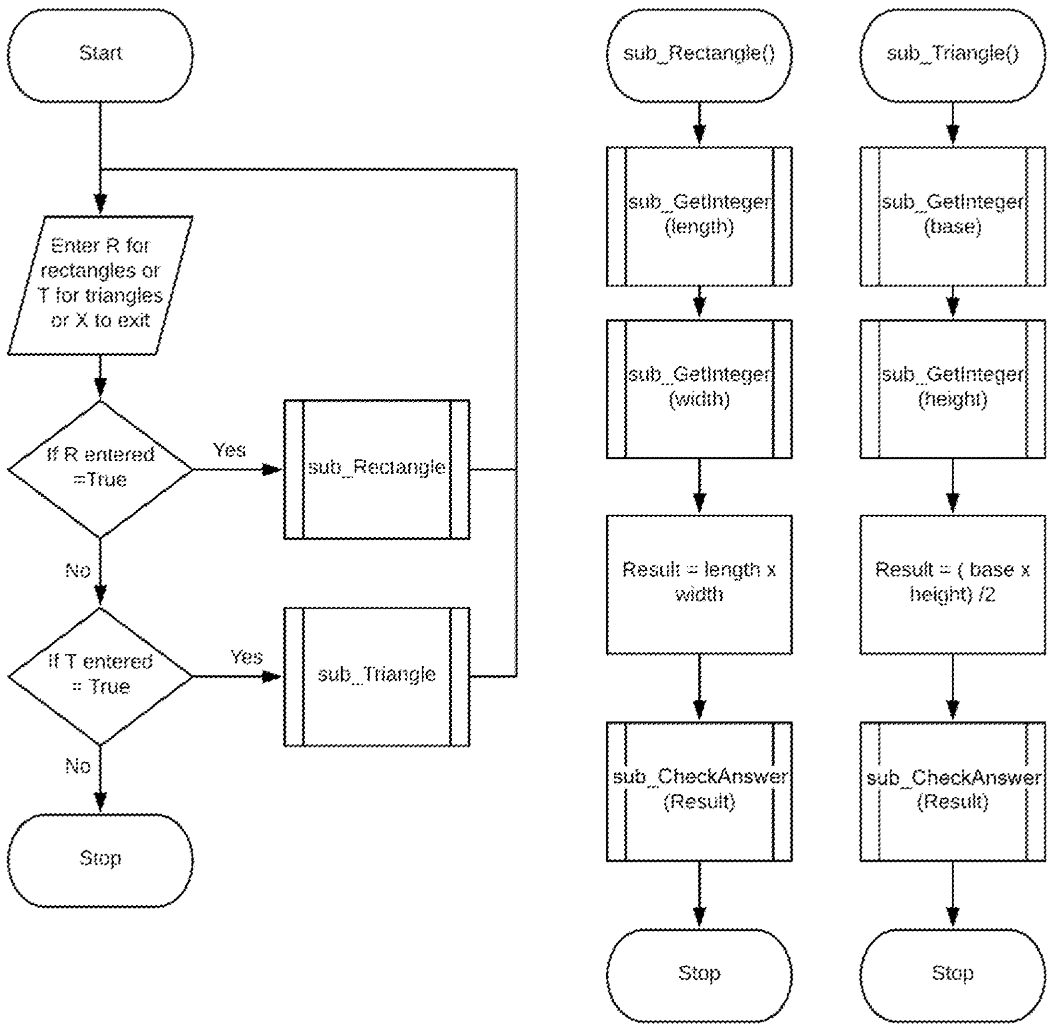
This is a p
a pla

T
ACT

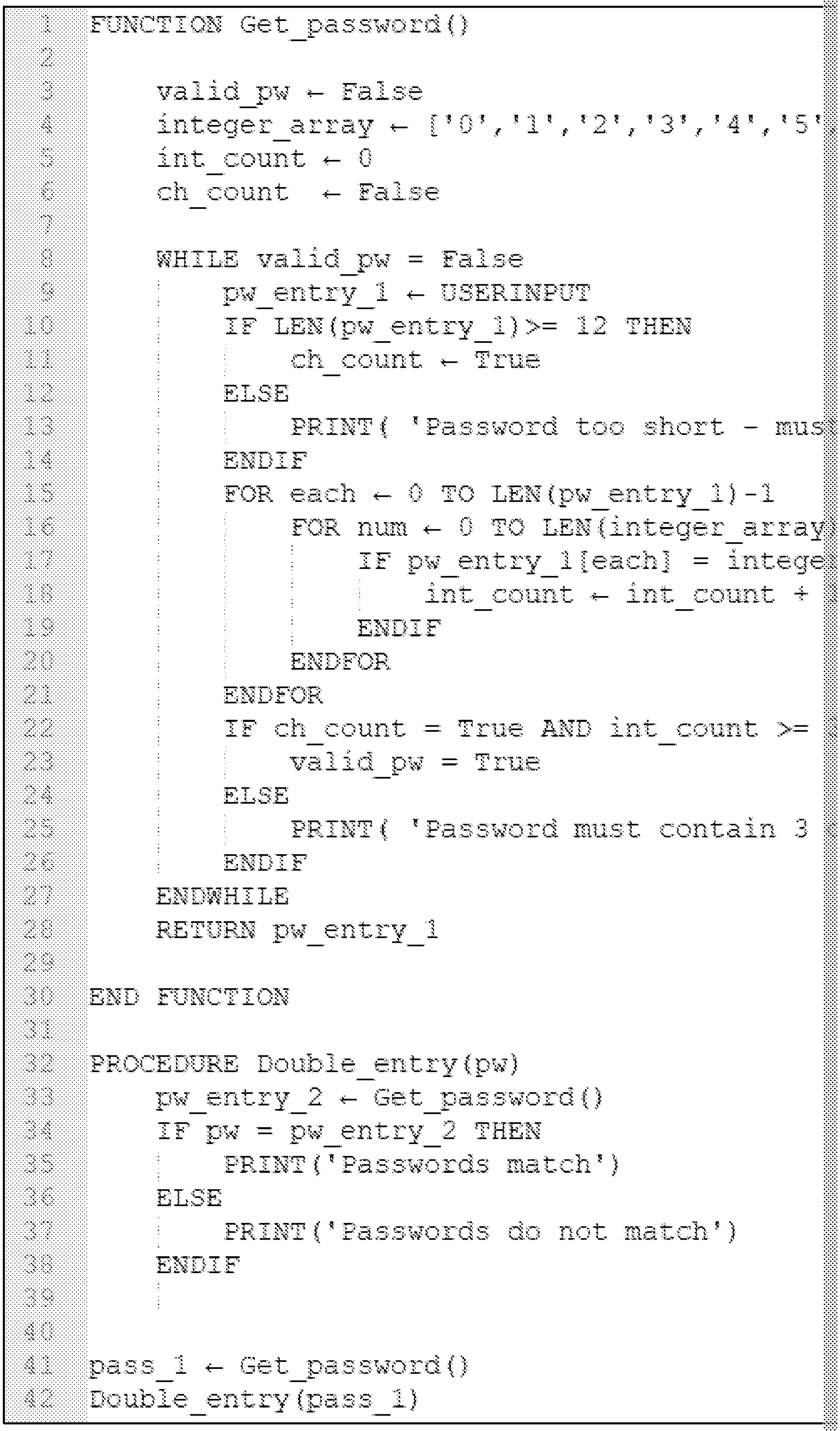
Exercise 17


Start
Enter R for rectangles or T for triangles or X to exit
If R entered =True
Yes
sub_Rectangle
No
If T entered = True
Yes
sub_Triangle
No
Stop
sub_Rectangle
Enter length
Enter width
R_Area = length x width
count = 0
Enter area
Incorrect Try again
R_Area = area?
No
count = count +1
is count =3?
No
Yes
Yes
Try another
The answer is + R_Area
Stop

Alternative Solution

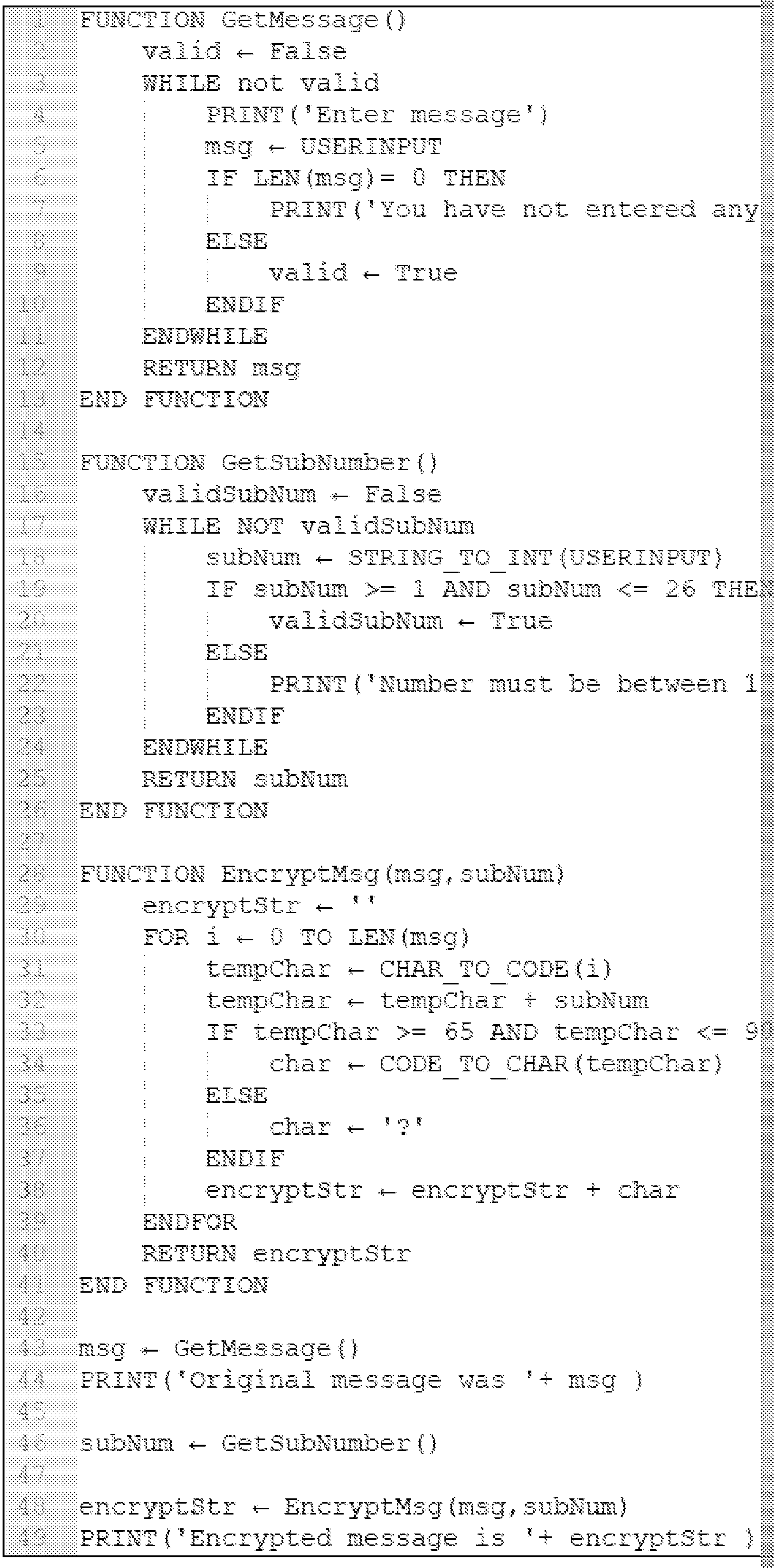


Exercise 18



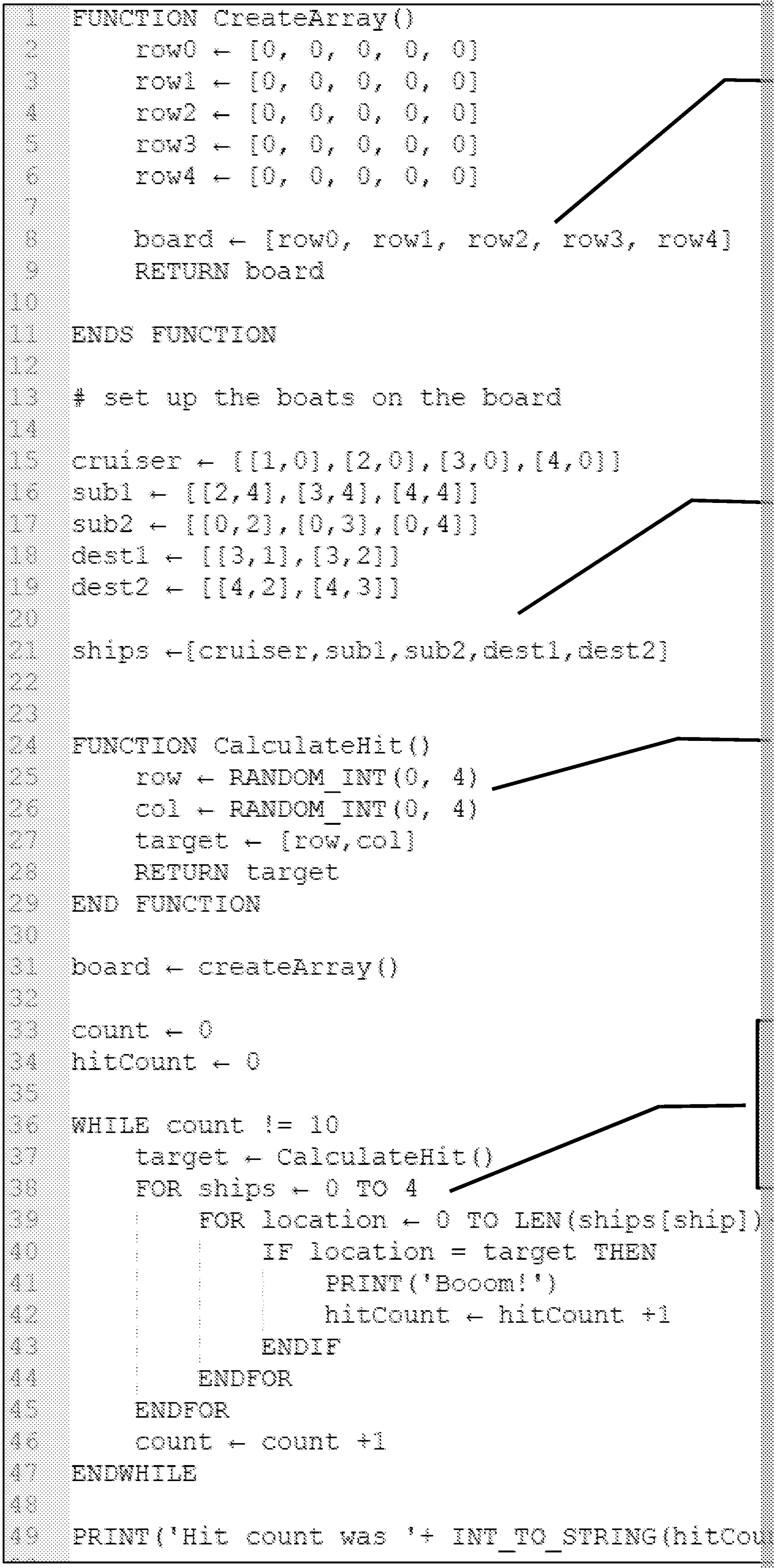
```
FUNCTION Get_password()

    valid_pw ← False
    integer_array ← ['0','1','2','3','4','5',
    int_count ← 0
    ch_count  ← False

    WHILE valid_pw = False
        pw_entry_1 ← USERINPUT
        IF LEN(pw_entry_1)>= 12 THEN
            ch_count ← True
        ELSE
            PRINT( 'Password too short - mus
        ENDIF
        FOR each ← 0 TO LEN(pw_entry_1)-1
            FOR num ← 0 TO LEN(integer_array
                IF pw_entry_1[each] = intege
                    int_count ← int_count +
                ENDIF
            ENDFOR
        ENDFOR
        IF ch_count = True AND int_count >=
            valid_pw = True
        ELSE
            PRINT( 'Password must contain 3
        ENDIF
    ENDWHILE
    RETURN pw_entry_1

END FUNCTION

PROCEDURE Double_entry(pw)
    pw_entry_2 ← Get_password()
    IF pw = pw_entry_2 THEN
        PRINT('Passwords match')
    ELSE
        PRINT('Passwords do not match')
    ENDIF


pass_1 ← Get_password()
Double_entry(pass_1)
```

Exercise 19



```
FUNCTION GetMessage()
    valid ← False
    WHILE not valid
        PRINT('Enter message')
        msg ← USERINPUT
        IF LEN(msg)= 0 THEN
            PRINT('You have not entered any
        ELSE
            valid ← True
        ENDIF
    ENDWHILE
    RETURN msg
END FUNCTION

FUNCTION GetSubNumber()
    validSubNum ← False
    WHILE NOT validSubNum
        subNum ← STRING_TO_INT(USERINPUT)
        IF subNum >= 1 AND subNum <= 26 THEN
            validSubNum ← True
        ELSE
            PRINT('Number must be between 1
        ENDIF
    ENDWHILE
    RETURN subNum
END FUNCTION

FUNCTION EncryptMsg(msg,subNum)
    encryptStr ← ''
    FOR i ← 0 TO LEN(msg)
        tempChar ← CHAR_TO_CODE(i)
        tempChar ← tempChar + subNum
        IF tempChar >= 65 AND tempChar <= 9
            char ← CODE_TO_CHAR(tempChar)
        ELSE
            char ← '?'
        ENDIF
        encryptStr ← encryptStr + char
    ENDFOR
    RETURN encryptStr
END FUNCTION

msg ← GetMessage()
PRINT('Original message was '+ msg )

subNum ← GetSubNumber()

encryptStr ← EncryptMsg(msg,subNum)
PRINT('Encrypted message is '+ encryptStr )
```

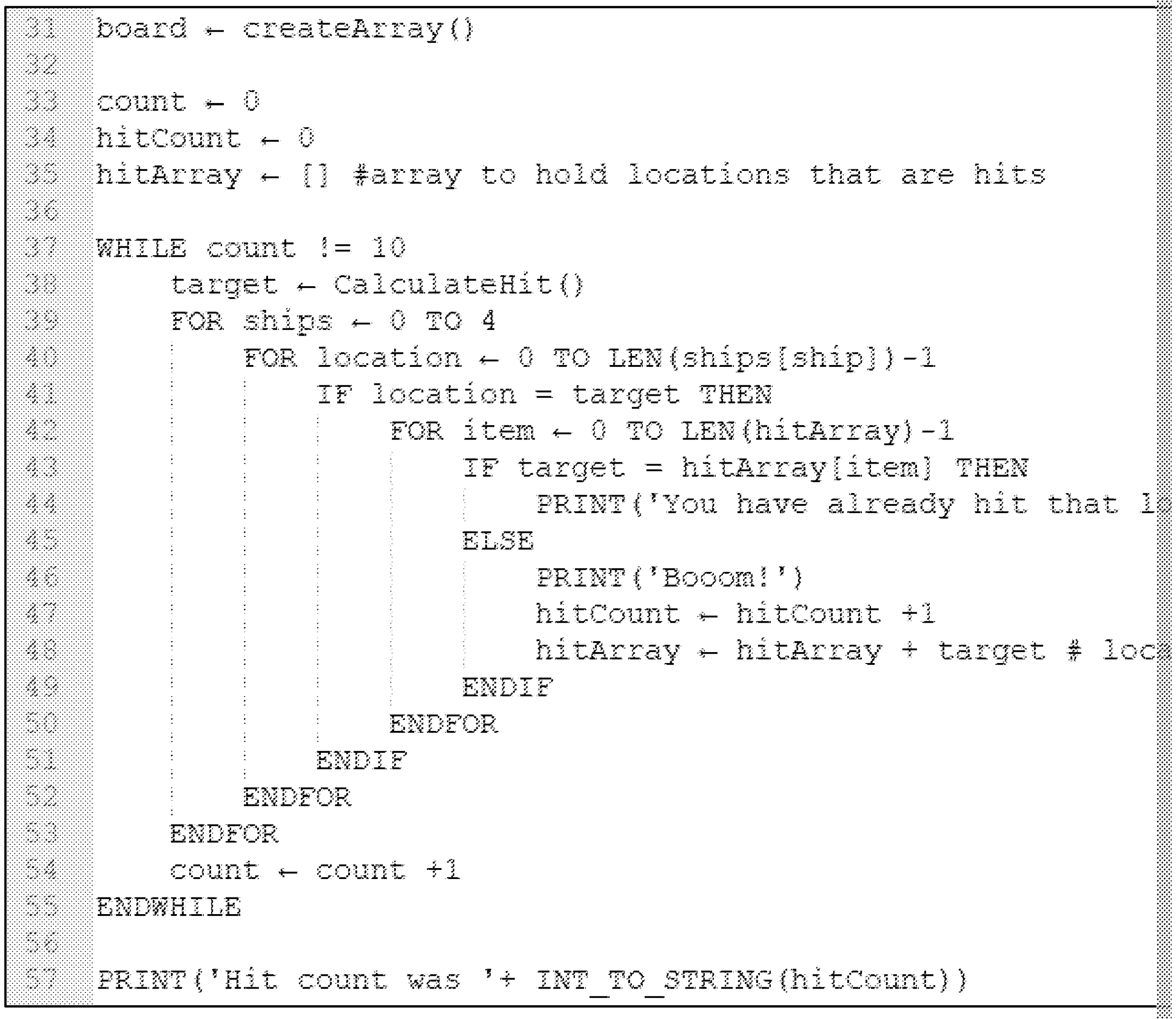
Exercise 20



```
FUNCTION CreateArray()
    row0 ← [0, 0, 0, 0, 0]
    row1 ← [0, 0, 0, 0, 0]
    row2 ← [0, 0, 0, 0, 0]
    row3 ← [0, 0, 0, 0, 0]
    row4 ← [0, 0, 0, 0, 0]

    board ← [row0, row1, row2, row3, row4]
    RETURN board

ENDS FUNCTION

# set up the boats on the board

cruiser ← [[1,0],[2,0],[3,0],[4,0]]
sub1 ← [[2,4],[3,4],[4,4]]
sub2 ← [[0,2],[0,3],[0,4]]
dest1 ← [[3,1],[3,2]]
dest2 ← [[4,2],[4,3]]

ships ←[cruiser,sub1,sub2,dest1,dest2]


FUNCTION CalculateHit()
    row ← RANDOM_INT(0, 4)
    col ← RANDOM_INT(0, 4)
    target ← [row,col]
    RETURN target
END FUNCTION

board ← createArray()

count ← 0
hitCount ← 0

WHILE count != 10
    target ← CalculateHit()
    FOR ships ← 0 TO 4
        FOR location ← 0 TO LEN(ships[ship])
            IF location = target THEN
                PRINT('Booom!')
                hitCount ← hitCount +1
            ENDIF
        ENDFOR
    ENDFOR
    count ← count +1
ENDWHILE

PRINT('Hit count was '+ INT_TO_STRING(hitCou
```

Exercise 20A

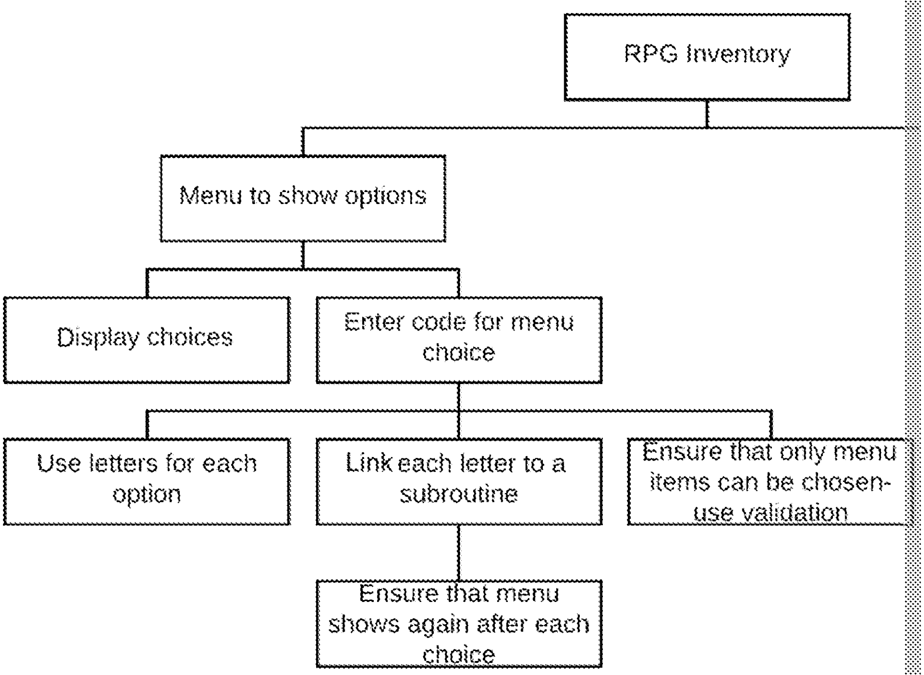
Note: The solution is to add each location that is a ‘hit’ into an array. Each time another location matches the targ to see whether that location has already been hit, and a message is displayed. If the location is not already in the



```
board ← createArray()

count ← 0
hitCount ← 0
hitArray ← [] #array to hold locations that are hits

WHILE count != 10
    target ← CalculateHit()
    FOR ships ← 0 TO 4
        FOR location ← 0 TO LEN(ships[ship])-1
            IF location = target THEN
                FOR item ← 0 TO LEN(hitArray)-1
                    IF target = hitArray[item] THEN
                        PRINT('You have already hit that l
                    ELSE
                        PRINT('Booom!')
                        hitCount ← hitCount +1
                        hitArray ← hitArray + target # loc
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    ENDFOR
    count ← count +1
ENDWHILE

PRINT('Hit count was '+ INT_TO_STRING(hitCount))
```

Exercise 21

Suggested plan for decomposing problem:


RPG Inventory
Menu to show options
Display choices
Enter code for menu choice
Use letters for each option
Link each letter to a subroutine
Ensure that only menu items can be chosen- use validation
Ensure that menu shows again after each choice

```
#runs the choices
PROCEDURE makeInventoryChoice(arr)
      menuOpt ← DisplayMenu()

      WHILE menuOpt != 'X'
            IF menuOpt = 'D' THEN
                  ViewInventory(arr)
                  makeInventoryChoice(arr) # shows t
            ELSE IF menuOpt = 'A' THEN
                  arr ← AddInventory(arr)
                  makeInventoryChoice(arr) # shows t
            ELSE IF menuOpt = 'U' THEN
                  arr ← UseInventoryItem(arr)
                  makeInventoryChoice(arr) # shows t
            ENDIF
      ENDWHILE

      ExitInventory()

END PROCEDURE

#display menu
FUNCTION DisplayMenu()
      PRINT('Enter D to view inventory')
      PRINT('Enter A to add to inventory')
      PRINT('Enter U to use an inventory item')
      PRINT('Enter X to exit inventory menu')

      menuChoice ← ['D','A','U','X']
      validChoice ← False

      WHILE NOT validChoice
            menuOpt ← USERINPUT
            FOR i ← 0 TO LEN(menuChoice)-1
            IF menuOpt = menuChoice[i] THEN
                validChoice ← True
            ELSE IF i = LEN(menuChoice)-1
                PRINT('Please enter a valid m
            ENDIF
        ENDFOR
    ENDWHILE
    RETURN menuOpt
END FUNCTION

#View inventory
PROCEDURE ViewInventory(arr)
    FOR i ← 0 TO LEN(arr)-1
        PRINT( arr[i])
    ENDFOR
END PROCEDURE

#Add item to inventory
FUNCTION AddInventory(arr)
    PRINT( 'Name item to be added')
    item ← USERINPUT
    arr ← arr + item
    RETURN arr
END FUNCTION

#Use an inventory item
FUNCTION UseInventoryItem(arr)
    notFound ← False
    PRINT( 'What item do you want to use?')
    item ← USERINPUT
    FOR i ← 0 TO LEN(arr)-1
        IF item != arr[i] THEN
            notFound ← True
            IF notFound THEN
                PRINT('The item is not in the
            ENDIF
        ELSE
            PRINT( 'You have now used this ite
            arr ← arr-[item]
        ENDIF
    ENDFOR
    RETURN arr
END FUNCTION

#Exit inventory menu
PROCEDURE ExitInventory()
    PRINT('You have exited the inventory menu')
END PROCEDURE

# call subroutines to run inventory
inventoryArray ← []
makeInventoryChoice (inventoryArray)
```

Fox, chicken and grain problem

You must take the chicken across the river with you first.

A	B
FG	C

Next, take the fox across, leave it there and return with the chicken.

A	B
CG	F

Next, take the bag of grain across and leave it with the fox.

A	B
C	FG

Finally, return and take the chicken across.

A	B
	FCG

Exercise 22

You would think that the quickest way is to have Adam (1) carry the torch and do all the achieved by having Clair (5) and Danni (10) cross together.

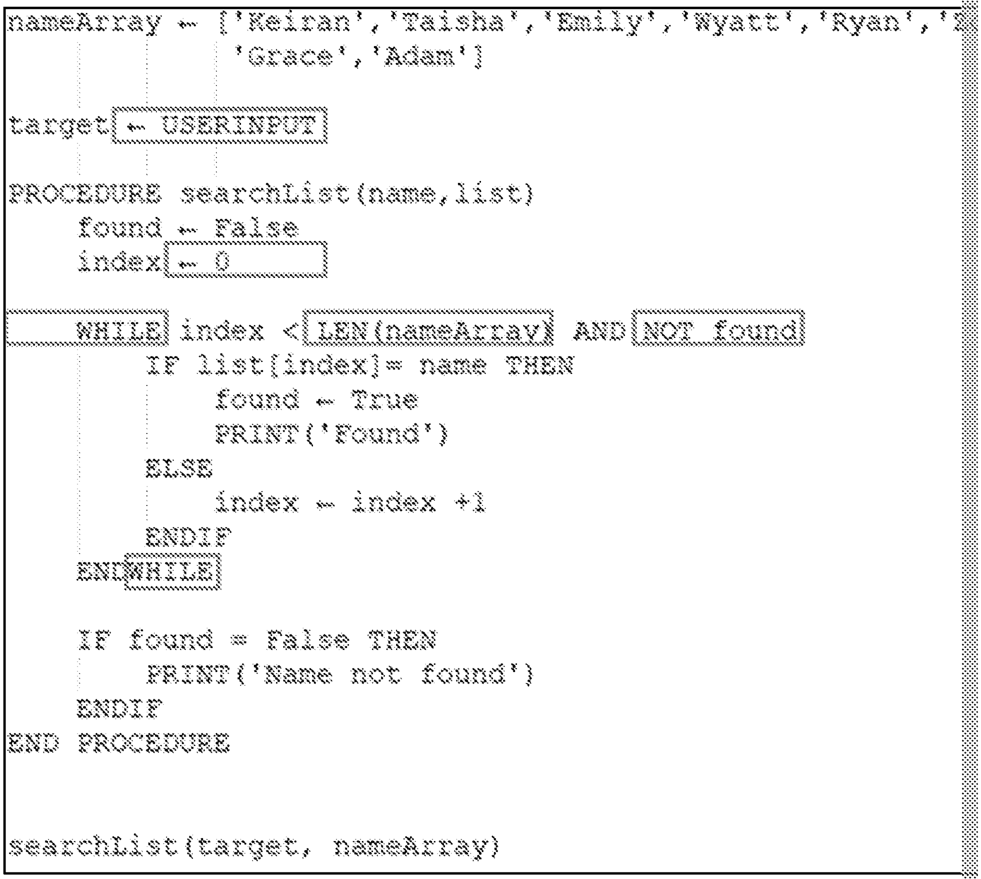
To simplify the solution, think about it like this first:

- A = 1
- B = 2
- C = 5
- D = 8

The moves are as follows:

Island	Bridge	Stage	Time Tal
C and D	A and B (with torch)	A and B	
A, C and D	A returns (with torch)	B	
A	C and D (with torch)	B, C and D	
A, B	B returns (with torch)	C and D	
	A and B (with torch)	A, B, C and D	
		TOTAL	

Exercise 23



```
nameArray ← ['Keiran','Taisha','Emily','Wyatt','Ryan','
                'Grace','Adam']

target ← USERINPUT

PROCEDURE searchList(name,list)
    found ← False
    index ← 0

    WHILE index < LEN(nameArray) AND NOT found
        IF list[index]= name THEN
            found ← True
            PRINT('Found')
        ELSE
            index ← index +1
        ENDIF
    ENDWHILE

    IF found = False THEN
        PRINT('Name not found')
    ENDIF
END PROCEDURE


searchList(target, nameArray)
```

Exercise 24

Linear search 1:

index	found	target	output
0	*False*	*34*	
1			
2			
3	True		Found at 3
4			
5			
6			
7			
8			
9			

Linear search 2:

index	found	target	
0	*False*	*1*	
1			
2			
3	True		F

Which is most efficient, and why?

Linear search 1 is less efficient as the FOR loop (FOR index ← 0 TO LEN (numsList)) contin when the search item has been found.

Linear search 2 uses a WHILE loop to check two conditions: whether the end of the array search item remains not found. The WHILE loop will only continue while BOTH conditions search stops as soon as the item has been found.

Exercise 25

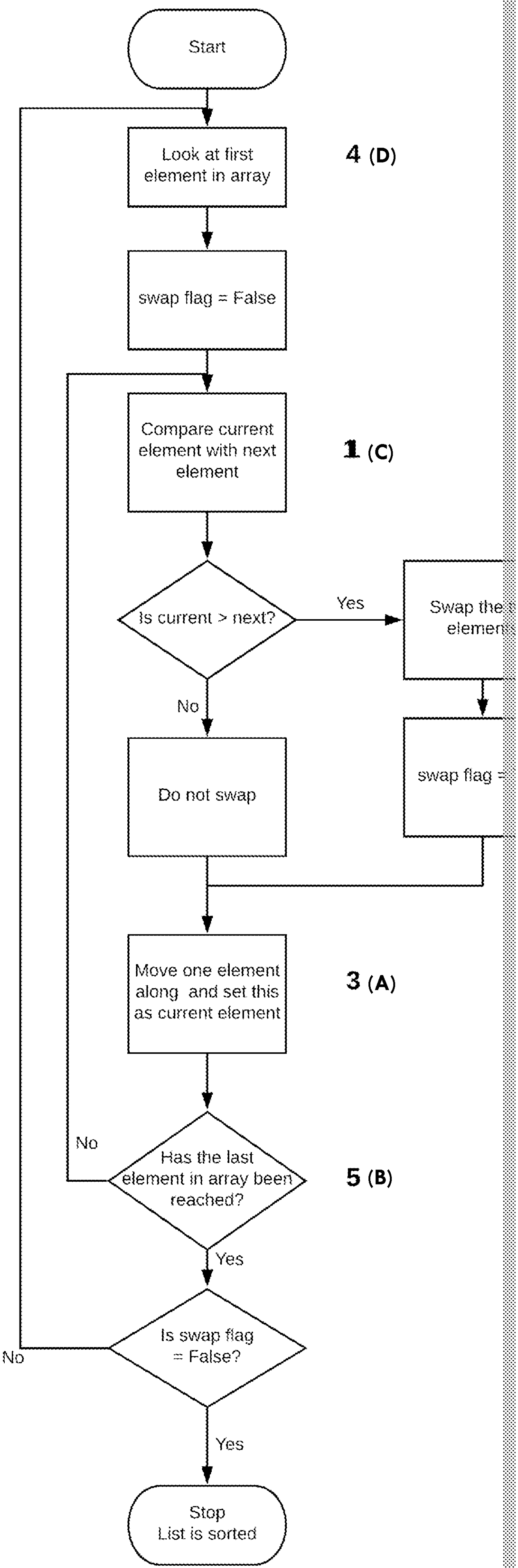
1.

5	*1*	*6*	*2*	*4*	*3*
1	5	6	2	4	3
1	5	2	6	4	3
1	5	2	4	6	3
1	5	5	4	3	6
1	2	5	4	6	3
1	2	4	5	3	6
1	2	4	3	5	6
1	2	3	4	5	6
1	2	3	4	5	6

Note: The final pass must be completed to confirm that no more swaps are needed

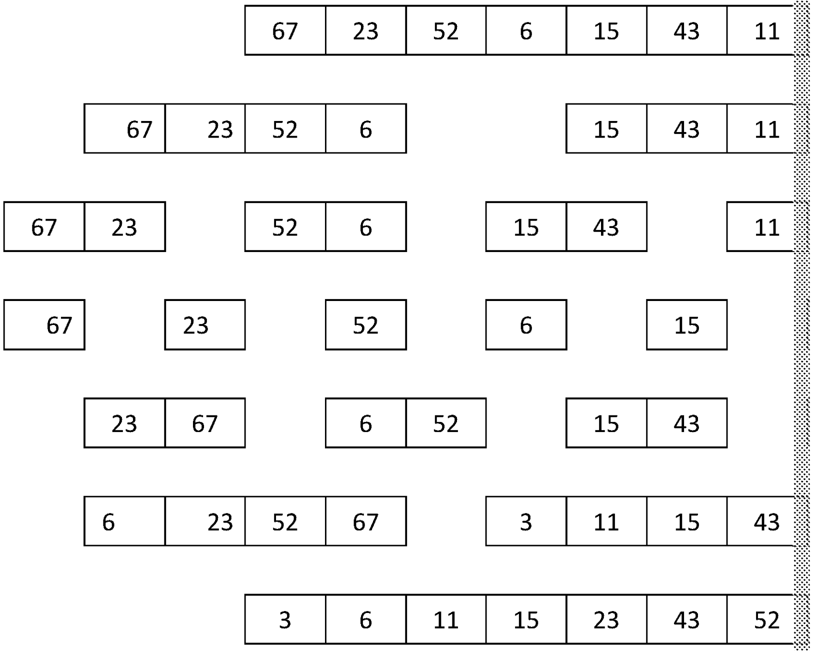
2.
1. *Compare the first two elements in the array.*
2. *Is the first element bigger than the second element?*
3. If the answer is yes, the elements are swapped.
4. Move forward by one element and compare the current element with the one
5. Repeat steps 2, 3 and 4 until the end of the array is reached.
6. Repeat steps 1 to 6 until no swaps have been made.

Exercise 26


Start
Look at first element in array
4 (D)
swap flag = False
Compare current element with next element
1 (C)
Is current > next?
Yes
Swap the elemen
No
Do not swap
swap flag =
Move one element along and set this as current element
3 (A)
No
Has the last element in array been reached?
5 (B)
Yes
No
Is swap flag = False?
Yes
Stop
List is sorted

Exercise 27

1. (a)



67 23 52 6 15 43 11

67 23 52 6 | 15 43 11

67 23 | 52 6 | 15 43 | 11

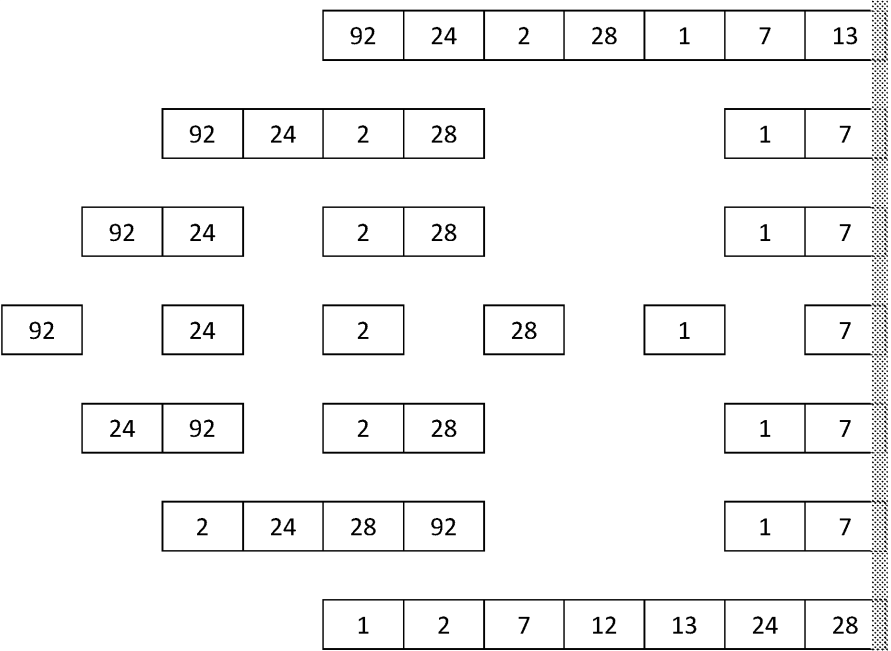
67 | 23 | 52 | 6 | 15

23 67 | 6 52 | 15 43

6 23 52 67 | 3 11 15 43

3 6 11 15 23 43 52

(b)



92 24 2 28 1 7 13

92 24 2 28 | 1 7

92 24 | 2 28 | 1 7

92 | 24 | 2 | 28 | 1 | 7

24 92 | 2 28 | 1 7

2 24 28 92 | 1 7

1 2 7 12 13 24 28

2. a) Linear search, as the array is unordered.
   b) The algorithm takes in an array of data and a search term 'n'. The algorithm th sequentially to see if it matches the search term. When the whole array has be the variable **found** as True if the search term is in the array, or False if it is not
   c) Line 6 could be edited to incorporate the Boolean logical AND as follows: **WHI False**. This will make the algorithm more efficient as the WHILE loop will finish found.

3.
- Compare items [0] and [1] to see which is larger.
- Swap items so item [0] is smaller than item [1].
- Continue the comparison between item [1] and item [2].
- Swap the items so that item [1] is smaller than item [2].
- Repeat the process until the end of the array.
- Return to the start of the array and repeat again until no swaps are made.

4. The merge sort is a ‘divide and conquer’ algorithm which divides an unsorted array only contains one value before sorting and merging each pair, set of four, etc. It is a for very large data sets as it uses this division method. However, it requires exactly locations as the size of the data to perform the sort so it is very inefficient in terms

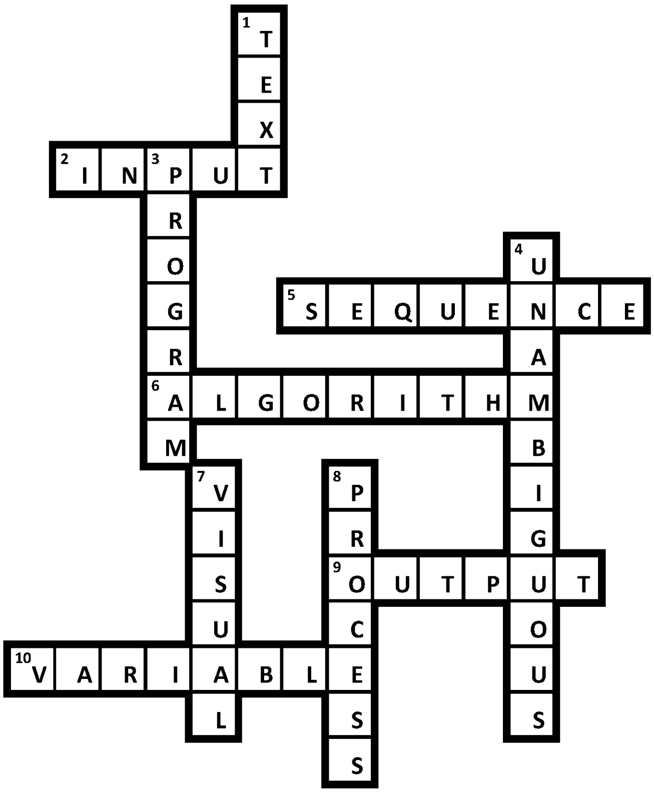
The bubble sort is very efficient in its use of memory, only requiring one memory lo being swapped. Unlike the merge sort, the bubble sort works by comparing each pa the comparisons and swaps increases rapidly as the size of the data increases, mak time it takes.

5. The algorithm takes in an array of data as its parameter, starting at the first item in item. The algorithm then compares each item in the array with this initial value to c value of variable **smallest** is changed to the smaller value. When it has compared al smallest value.

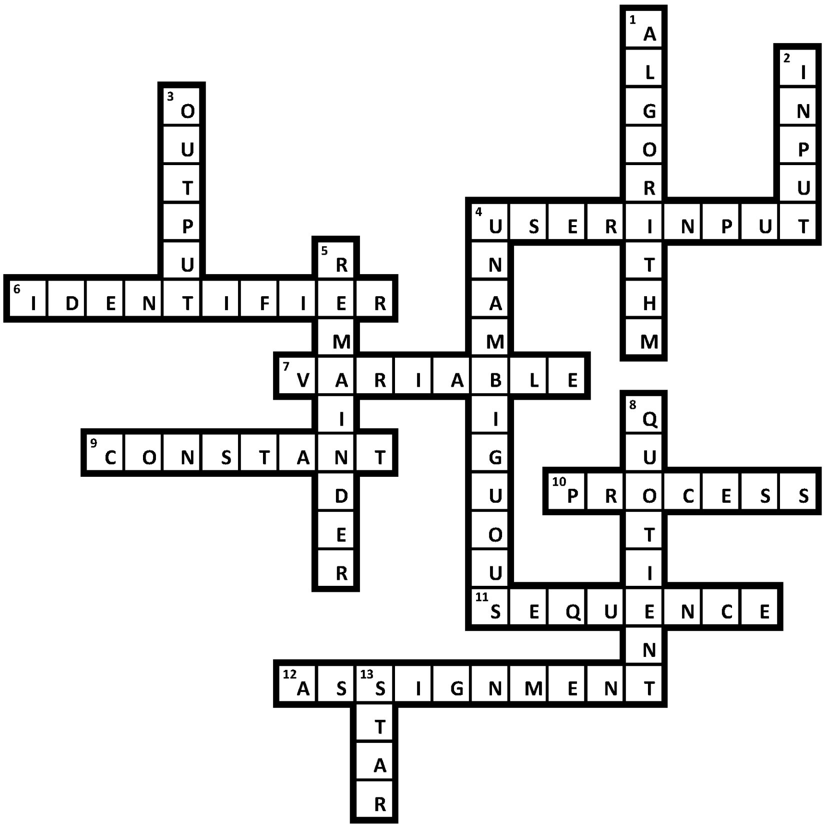
6. A binary search would be most suitable since the array is already sorted. A linear se very small and the time difference would be very small.

Crosswords

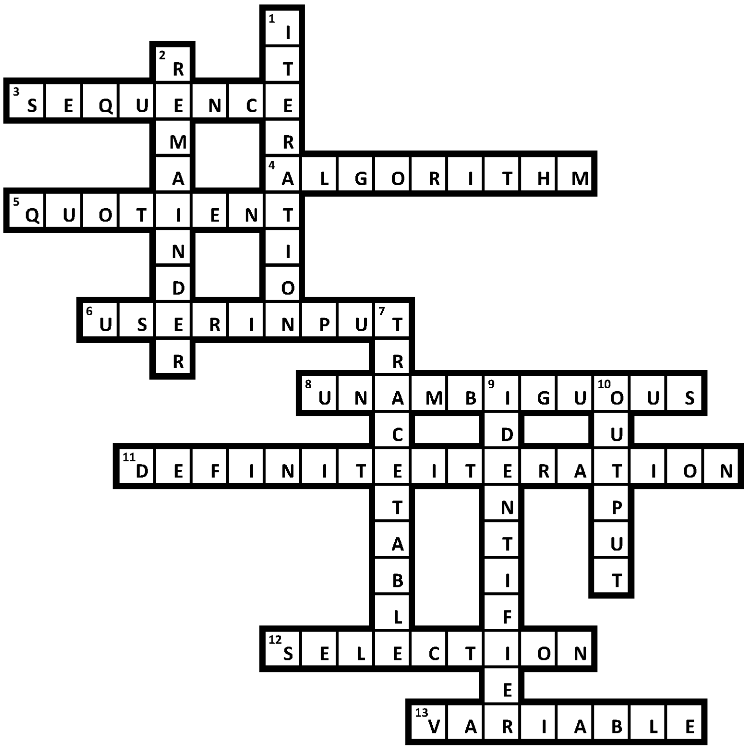
Crossword 1



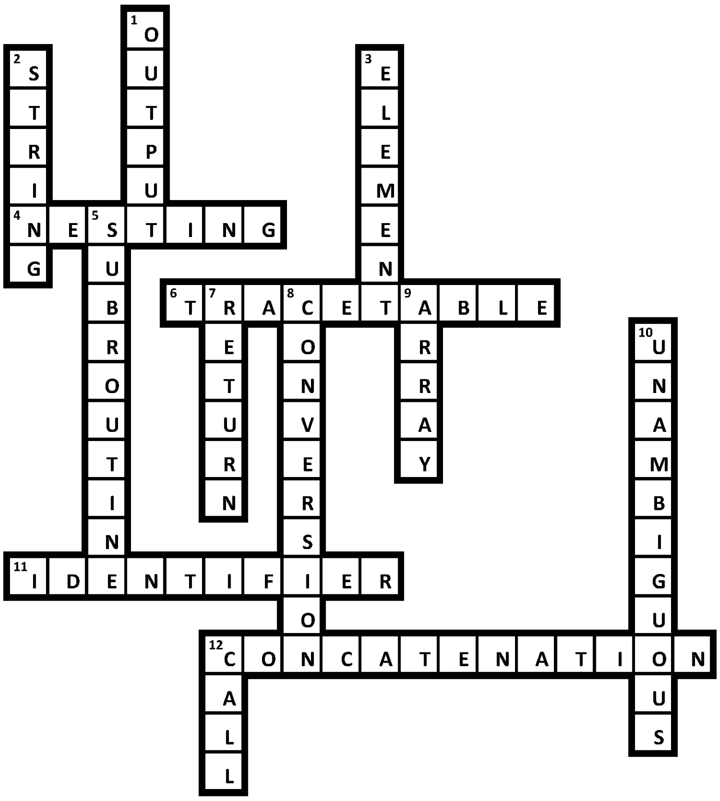
Crossword 2



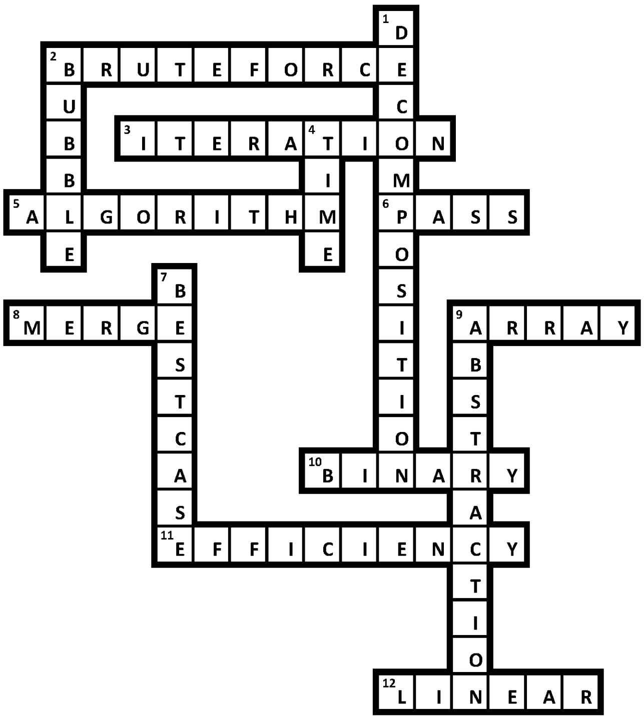
Crossword 3



Crossword 4



Crossword 5


1 DECOMPOSITION
2 BRUTEFORCE
BUBBLE
3 ITERATION
4 TIME
5 ALGORITHM
6 PASS
7 BESTCASE
8 MERGE
9 ARRAY
ABSTRACTION
10 BINARY
11 EFFICIENCY
12 LINEAR