# Worksheet 5 Subroutines Answers

**Task 1**

1. A primary school teacher requires a program that will allow pupils to practise their multiplication tables (times tables). The program must allow them to choose the table they want displayed and the start and end numbers to multiply by.

 For example, if the pupil enters 5, 4, 12 the program will display

 5 x 4 = 20

 5 x 5 = 25

 …

 …

 5 x 12 = 60

 The program will then print a message followed by the table selected.

 Create a pseudocode solution for this using a subroutine called **multiples()** which takes **table**, **startnum**, **endnum** and **pupilName** as parameters. The subroutine will output the message and multiplication table. The main program will prompt the user to enter the values and will then pass them to the routine.



SUB multiples(f1,f2start,f2end,name)

 OUTPUT “Hi”,name, “ here is your times table”

 FOR index 🡨 f2start to f2end

 OUTPUT (f1, “ x ”, index, “ = ”, f1 \* index)

 END FOR

ENDSUB

#main program

OUTPUT “What is your name? ”

name 🡨 USERINPUT

OUTPUT “Enter times table, start number and end number ”

table 🡨 USERINPUT

startnum 🡨 USERINPUT

endnum 🡨 USERINPUT

multiples(table,startnum,endnum,pupilName)

**Task 2**

2. It is possible to return more than one value from a function. Consider the following Python program:

# pass multiple results back from a function

def calc(a,b)

 constantVal = 2

 x = constantVal \*(a + b)

 y = a - b

 z = a \* b

 return x, y, z

#main program

add, subtract, multiply = calc(5,3)

OUTPUT add, subtract, multiply

 (a) Name a **local variable** in the subroutine constantVal

 (b) What is printed by the statement in the final line of the program? 16 2 15

3. It is often necessary to verify information upon data input. One method of doing this is double entry.

 Write pseudocode for a subroutine called **getPword()** that takes one parameter, called **attempt**, which can have a value of 1 or 2. The subroutine should prompt the user to enter a password if **attempt = 1**, or prompt the user to re-enter a password if **attempt = 2**, and then return the password.

 The subroutine should also check that the length of the password is a valid length between 6 to 8 characters. The main program will verify that the two passwords are the same, and re-prompt for entry if they are not. If both passwords are the same, a message is displayed, informing the user that the password change has been successful.

 The output below demonstrates how the program will work.



SUB getPword(attempt)

 pword 🡨 “”

 while len(pword) < 5 or len(pword) > 8

 IF attempt = 1 THEN

 OUTPUT “Enter password”

 ELSE

 OUTPUT “Enter password again”

 END IF

 pword 🡨 USERINPUT

 if len(pword) < 5 or len(pword) > 8

 OUTPUT “Error. Password must be 6 to 8 characters long”

 END WHILE

 RETURN pword

ENDSUB

#main program

pword1 🡨 getPword(1)

pword2 🡨 getPword(2)

WHILE pword1 <> pword2

 OUTPUT “Error passwords do not match”

 pword1 🡨 getPword(1)

 pword2 🡨getPword(2)

END WHILE

OUTPUT “Password change successful”

**The next task builds on Question 6 in Worksheet 4.**

4. A company runs a private car park near an airport. The car park has 10 rows numbered 1-10 and each row has spaces (referred to as columns) numbered 1-6 for 6 cars. Customers leave their cars with keys at the car park office, and a driver parks it in a free space.

 The space is referenced by its grid coordinates row and column. E.g. a car parked in the 3rd row, 5th space would have the grid reference [3,5].

 The driver enters the car registration into the computer. A car with registration AVH 61 HU parked at grid reference [3,5] would assign “AVH 61 HU” to **park[3,5]**. Empty spaces are denoted, for example, by **park[3,5] = “empty”**

 Write pseudocode for a program which displays a menu with 5 options, and for the first four options, calls the relevant subroutine.

 Option 1: Set all spaces in the car park to “empty”

 Option 2: Park a car. This option asks the user to enter the registration number of a car and the grid reference (row and column number) where it has been parked. The program checks that this is an empty space, and if it is, puts the registration number in the appropriate element of the array. If it is not, it asks the user to enter the grid reference.

 Option 3: Remove a car. This option asks the user to enter the registration number, searches the grid for the number and then resets it to “empty”.

 Option 4: Display the car park grid

 Option 5: Quit

 Pseudocode for the main program is given below. This initialises the car park grid to “empty” and then repeatedly displays the menu of options and performs the required function until the user selects “Quit”.

 Write subroutines for options 2-5. (Assume array indices start at 0)

 Note that the way the 2-D array is initialised will vary in different programming languages – the pseudocode does not reflect this. Program solutions in Python and VB are provided.

 In Python, for example, the array needs to be initialised before it is passed as a parameter to the subroutines.

 #main program

 initialise car park grid to “empty”

#display menu of options

OUTPUT ("1. Reset all spaces in the car park to 'empty'")

OUTPUT ("2. Park a car")

OUTPUT ("3. Remove a car")

OUTPUT ("4. Display the car park grid")

OUTPUT ("5. Quit\n")

option 🡨 USERINPUT("Enter your choice: ")

#accept choice

while option <> "5":

 if option = "1"

 emptyCarPark(carPark)

 else if option = "2"

 parkACar(carPark)

 else if option = "3"

 removeACar(carPark)

 else if option = "4"

 displayCarParkGrid(carPark)

 else

 option 🡨 ("Invalid choice - please re-enter: ")

 end if

 OUTPUT ("1. Reset all spaces in the car park to ‘empty’")

 OUTPUT ("2. Park a car")

 OUTPUT ("3. Remove a car")

 OUTPUT ("4. Display the car park grid")

 OUTPUT ("5. Quit\n")

 option 🡨 USERINPUT("Enter your choice: ")

end while

OUTPUT ("Goodbye")

SUB emptyCarPark(carpark)

 #populate car park 2-d array with “empty” in each grid reference

 FOR row 🡨 0 TO 9

 FOR column 🡨 0 TO 5

 carPark[row][col]=("empty")

 NEXT column

 NEXT row

 OUTPUT("car park is now empty")

ENDSUB

SUB parkACar(carpark)

prompt for and enter car registration

prompt for and enter row and column where car is parked

emptySpace 🡨 False

WHILE emptySpace = False

 WHILE row < 1 OR row > 10

 row 🡨 USERINPUT("Row must be between 1 and 10 -
 please re-enter: ")

 ENDWHILE

 row 🡨 row-1

 rowValid 🡨 True

 WHILE column < 1 or column > 6

 column 🡨 USERINPUT("Column must be between 1 and 6 -
 please re-enter: ")

 ENDWHILE

 column 🡨 column - 1

 columnValid 🡨 True

 IF rowValid and columnValid and carPark[row][column]= "empty" THEN

 emptySpace 🡨 True

 ELSE

 rowValid 🡨 False

 columnValid 🡨 False

 row 🡨 100

 column 🡨 100

 OUTPUT ("That space is taken”)

 ENDIF

ENDWHILE

carPark[row][column] 🡨 regNo

ENDSUB

SUB RemoveACar(carpark)

 OUTPUT("Enter the car registration: ")

 regNo 🡨 USERINPUT

 carFound 🡨 False

#search for car

 FOR row 🡨 0 TO 9

 FOR column 🡨 0 TO 5

 symbol 🡨 carPark[row][column]

 IF symbol = regNo

 carPark[row][column] 🡨 "empty"

 carFound 🡨 True

 ENDIF

 NEXT column

 NEXT row

 IF carFound = False

 OUTPUT("Car not found")

ENDSUB

SUB displayCarParkGrid(carpark)

FOR row 🡨 0 TO 9

 FOR column 🡨 0 TO 5

 symbol 🡨 carPark[row][column]

 WRITE (symbol,' ',) #print without moving to new line

 NEXT column

 WRITELINE #move to new line

NEXT row

ENDSUB