# 1. Describe briefly three advantages of using subroutines in programs. [6]

Helps make programs efficient, since if you want to repeat a block of code multiple times in a program, instead of having to type out the piece of code manually every time, you can just copy and paste the subroutine name (call the subroutine) and immediately get the program to run that piece of code. Saves a lot of time because you don’t have to be wasting time repeatedly typing out the same block of code.

Another advantage of subroutines is that it makes the program’s code more manageable. If you call a subroutine and something in the program goes wrong when the subroutine is being run, you will know immediately that the error is in the subroutine making it so much easier to check for errors since if you didn’t use a subroutine and manually typed out every block of code that needs to be repeated, there is a good chance at some point you made some errors like syntax or logic errors, which means if there is a bug, you have to look through every single block of code for the error whereas using a subroutine, you don’t have to since the block of code inside the subroutine applies to everytime it is called so if something is wrong, fix the subroutine and that’s that.

One other advantage is that it makes the program’s code more readable for other developers. If you have a subroutine, it will have a name which should usually be a meaningful name. When read by other developers, they will be easily able to tell what the subroutine is for. If you don’t use a subroutine and choose to manually type out every block of code, other developers might not be able to understand it quickly because they have to spend more time seeing what each bit of your code does.

# 2. (a) What is a global variable? [1]

A variable that is declared in the main program and therefore can be used anywhere in the code it is declared in. It has the scope of the code.

# (b) What is a local variable? [1]

Local variables are variables declared not in a main program, but rather in a block of code like inside for loops, while loops etc. Their scope is only as far as the block of code they are declared in runs.

# (c) What is the advantage of using local variables in subroutines? [2]

# You wont accidentally change the value of a global variable outside the subroutine which can be troublesome if you want something to stay constant. Local variables only have the scope of the block of code they are declared in, they are self-contained.

# 3. Write a program in pseudocode that allows the user to add a name to a list array that holds up to ten names. The main program should call a subroutine that presents the user with a menu of three choices:

# 1 Add name

# 2 Display list

# 3 Quit

# The subroutine accepts and validates the user choice (1-3). If it is invalid, an appropriate error message should be displayed and the user asked to re-enter until they input a valid choice.

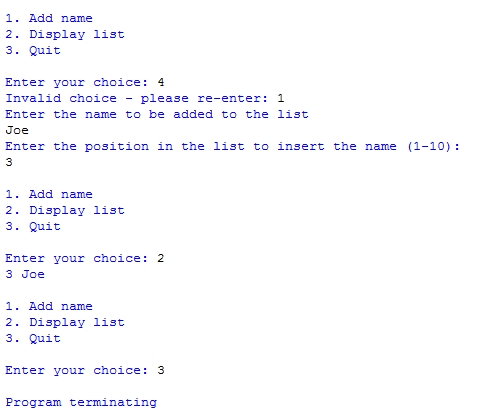
The subroutine is called with the statement

choice = displayMenu()

The main program then branches to one of two subroutines which adds a name or displays the list, or quits the program with a message “Program terminating” if the user selects option 3.

# If the user chooses to add a name, they should be prompted to enter a list number indicating where they want the name to be inserted. If the location is occupied, it will overwrite the name.

# The program will also provide an option to display the list. The output should look something like the example below. [10]



namesList = []

def displayMenu()

    presentChoices = None

    while True do

        try

            presentChoices = int(input("choose from 1-3: \n1. add name \n2. display name \n3. quit"))

            if presentChoices < 4 and presentChoices > 0 then

                break

        except ValueError

            print("invalid choice")

    if presentChoices == 1 then

        name = input("enter name to be added to list")

        position = int(input("enter which position you want the name to be in the list 1 to 10"))

        if position > len(namesList) then

            namesList.extend([None]) \* (position - len(namesList))

        namesList[position - 1] = name

    elif presentChoices == 2

        print(len(namesList), namesList)

    else

        print("program closing")

        return

endprocedure

choice = displayMenu()

print(choice)

[Total 20 marks]