**BSc (Hons) Computing**

**Module Code: QH0305     Module Title: Problem Solving**

# Assessment Sheet 6

## Instructions:

This is one of the eight assessment tasks, which will contribute to the overall marks. You will need to complete the tasks as outlined below and then document them in a Word document. As a minimum, you should provide screenshots of the following:

- Your code
- The output that your code generates

In instances where your code could give different outputs (depending on what values it is given), you should provide multiple screenshots of the console screen showing the different outputs to demonstrate that the code is working correctly.

This assessment sheet will focus on all the concepts you have learnt up until now.

**You must attempt all tasks on this sheet to achieve a higher grade. For example, if you want to gain Grade A, you must complete all other grades first and add them to your portfolio with screenshots.**

**A zip folder with all Grade codes must be attached inside of the portfolio (MS Word file).**

## Task 6:

One of the common tasks that computer programs are required to perform, is organising data into a required order. To do this, we need an algorithm. An algorithm is just a systematic solution to a problem. For this assessment task, we are going to consider a very simple sorting algorithm and you will need to complete the tasks as outlined below.

***Please note that there are many different sorting algorithms available.***

### Scenario:

You are working on a project to help a school organise a list of student names for an upcoming event. The school has received a list of student names, but they are all jumbled up. Your task is to implement a sorting algorithm to alphabetically order the student names.

The sort algorithm that you will implement in the example is a basic version of the Bubble Sort algorithm. Bubble Sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The process is repeated until the entire list is sorted. In each pass of the Bubble Sort algorithm, the largest element "bubbles" up to its correct position, hence the name Bubble Sort.

**To achieve a D grade**

Please follow the steps below:

- Create an array "**studentNames**" which stores the following names: "Emma", "Sophia", "Liam", "Noah", "Olivia", "Isabella", "Mia", "Charlotte", "Amelia".

- Implement a loop that iterates through the "**studentNames**" array.

- For each iteration of the loop, print out two names:
    - The name stored inside the array element that the loop is currently iterating through.
    - The name stored inside the adjacent element of the array.

- **Example output:**

  ```
  Student Name: Emma, Adjacent Student Name: Sophia
  Student Name: Sophia, Adjacent Student Name: Liam
  Student Name: Liam, Adjacent Student Name: Noah
  Etc.
  ```

- You need to provide screenshots of your code and the different outcomes it can give.
- Your Word document should have appropriate headings to ensure that this task can be easily identified alongside the rest of your work.

**To achieve a C grade**

Complete all previous steps, then modify the program to sort the array as follows:

- Update the loop to check if the current name being accessed is greater than the name stored in the adjacent position (use string comparison). If it is, swap the two names.

- Remove the print statements from the sorting loop.

- Print the "studentNames" array before and after the loop to verify the correct result. he student whose name comes last in alphabetical order should be at the end of the array. Note that the rest of the array may not be in the correct order.

- Print the " studentNames" array before and after the loop to verify the correct result.

  <u>Here is an example of how the algorithm works step by step:</u>

  Example of how the algorithm works step by step:

  Step 1: Before Sorting

  **Student Names: Carol, Eve, Bob, David, Alice**

Step 2: First Pass

Comparing "Carol" and "Eve": No swap ("Carol" < "Eve")

Comparing "Eve" and "Bob": Swap ("Eve" > "Bob")

Comparing "Eve" and "David": Swap ("Eve" > "David")

Comparing "Eve" and "Alice": Swap ("Eve" > "Alice")

**Student Names after the first pass: Carol, Bob, David, Alice, Eve**

(Statements in BOLD are printed on the console)

## To achieve a B grade

Complete all previous steps, then modify the program to sort the entire "studentNames" array using a nested loop as follows:

- Implement an additional loop that iterates through the "studentNames" array, comparing each name with its adjacent name, using string comparison.
- Swap the names if necessary to sort the array in ascending order.
- As before, you should print out the "studentNames" array before and after the sorting process to verify the correct result. Make sure that it works. Successful completion of this task will result in an array that is correctly ordered alphabetically.

All tasks must be accompanied by written descriptions or annotations. These must show satisfactory understanding of what the code is doing.

## To achieve an A grade

Complete all previous steps, then:

- In most assessment sheets, to attain a higher grade, tasks will require some independent research. There are many ways to sort an array of numbers. Investigate the different sorting algorithms out there and implement one of them (e.g., merge sort, insertion sort, quicksort, etc.). It should be different to the one you applied above.

- Note: Provide explanations in your Word document about the chosen algorithm, how it differs from bubble sort, and how you implemented it in your code.

- Remember to document your progress and provide screenshots, code, and explanations in the Word document as you move through different grades (D, C, B, A) of the assignment.

## Assignment Preparation Guidelines

- All components of the assignment report must be Word-processed **(handwritten text or hand drawn diagrams are not acceptable)**, font size must be within the range of 11 to 14 including the headings, body text and any texts within diagrams.
- Standard and commonly used fonts such as Times New Roman, Arial or Calibri should be used.
- All figures, graphs and tables must be numbered and labelled with short explanations.
- Material from external sources must be properly acknowledged and cited within the text using the Solent Harvard referencing system.
- All components of the assignment (text, diagrams, code etc.) must be submitted in one Word file.
- The report should be logically structured: the core of the report may start by defining the problem/requirements, followed by the proposed solution including a detailed discussion, analysis, and evaluation, leading to the implementation, and testing stage. Finally, there should be a conclusion and a personal reflection on learning.
- Screenshots without description / discussion are not suitable as they do not express your understanding or support your work adequately.

## Submission instructions

- This is a portfolio assignment with eight tasks in total. Each task will be completed and saved in the portfolio. Once the portfolio is completed, it should be submitted on Turnitin. The submission link to Turnitin can be found under the “Assessment Tab” in your module section in the SOL VLE.

- Please note file size limitation might apply. Your report must be under 250MB.

- The source code for each task should be **zipped** and **attached** to your Word document report submission in the appendix.

- The Assignment Brief can be found under “Assessment Tab” in your module section in the SOL VLE.

- **Refer to the Assignment Brief** to find the links to Late Submissions, Extenuating Circumstances, Academic Misconduct, Ethics Policy, Grade marking and Guidance for online submission through Solent Online Learning (SOL).