UNIVERSITY OF WESTMINSTER

School of Computer Science and Engineering
TIMED ASSESSMENT SEMESTER 2 2020/21

Module Code:	**5SENG001W, 5SENG002C**
Module Title:	**Algorithms: Theory, Design and Implementation**
Module Leader:	**Klaus Draeger**
Exam Start Time:	**Monday, 10th May 2021, 10:00 (BST)**
Recommended Exam End Time:	**Monday, 10th May 2021, 11:30 (BST)**
Submission Window:	**1h 10 minutes**
Submission Deadline:	**Monday, 10th May 2021, 12:40 (BST)**

Instructions to Candidates:

Please read the instructions below before starting the paper

- Module specific information is provided below by the Module Leader
- The Module Leader will be available during the exam release time to respond to any queries via the Discussion Board in the Assessment area of the module's Blackboard site
- As you will have access to resources to complete your assessment any content you use from external source materials will need to be referenced correctly. Whenever you directly quote, paraphrase, summarise, or utilise someone else's ideas or work, you have a responsibility to give due credit to that person. Support can be found at:

  https://www.westminster.ac.uk/current-students/studies/study-skills-and-training/research-skills/referencing-your-work
- This is an individual piece of work so do not collude with others on your answers as this is an academic offence
- Plagiarism detection software will be in use
- Where the University believes that academic misconduct has taken place the University will investigate the case and apply academic penalties as published in Section 10 Academic Misconduct regulations.
- ***Once completed please submit your paper via the Assignment content. In case of problems with submission, you will have TWO opportunities to upload your answers and the last uploaded attempt will be marked. Note that instructions on how to compile and submit your handwritten and/or typed solutions will have been sent to you separately.***
- ***Work submitted after the deadline will not be marked and will automatically be given a mark of zero***

Module Specific Information

PLEASE WRITE YOUR STUDENT ID CLEARLY AT THE TOP OF EACH PAGE

You are advised (but not required) to spend the first ten minutes of the examination reading the questions and planning how you will answer those you have selected.

Answer ALL questions in Section A and at least THREE questions from Section B.
Section A is worth a total of 40 marks.
Section B is worth a total of 60 marks.
Each question in Section B is worth 20 marks.
In Section B, only the THREE questions with the HIGHEST MARKS will count towards the FINAL MARK for the EXAM.

Section A

Answer ALL questions from this section.

Question 1

Consider this array representation of a min-heap:

index	0	1	2	3	4	5	6
value	2	6	3	7	8	9	4

1. Draw the tree representation of this heap. **[2 marks]**
2. Perform the following operations in the given order, drawing the updated representation after each operation:
   (a) Insert the number 5 **[2 marks]**
   (b) Insert the number 1 **[2 marks]**
   (c) Insert the number 0 **[2 marks]**
   (d) Pop the minimum off the heap **[2 marks]**

[TOTAL 10]

Question 2

Write a method that finds the middle of a doubly linked list. The DList class which contains the method has attributes `first` and `last` containing the first and last nodes of the list, respectively.
If the list has an odd number of nodes, the middle is a unique node, and the function should return that node's data.
If the list has an even number of nodes, the middle consists of two nodes, and the function should return the average of their values.
Complete either the Java or C++ version below.

- Java version:

```
public class DList{
 public class ListNode{
  public int data;
  public ListNode next;
  public ListNode previous;
 }
```

```
 public ListNode first, last;

 public int middle(){
  // TO DO
 }
}
```

- C++ version:

```
class DList{
 public:
 class ListNode{
  public:
  int data;
  ListNode *next;
  ListNode *previous;
 };

 ListNode *first, *last;

 int middle(){
  // TO DO
 }
};
```

[10 marks]
[TOTAL 10]

Question 3

For each of the following, state its best case and worst case execution time growth rates using the **Big 'O'** notation and give an informal justification for your answer:

- *Linear search* for a value in a linked list of values.
- *Binary search* for a value in an ordered array of values.
- *Bubblesort* for an array of values.

[10 marks]
[TOTAL 10]

Question 4

The following sets of *vertices* V and *edges* E represent a *directed graph*:

$$V = \{\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\ \}$$
$$E = \{\ (1,2),\ (2,3),\ (2,4),\ (3,5),\ (4,7),\ (5,4),\ (5,6),\ (6,7),\ (7,1)\ \}$$

Draw the graph represented by V and E above and give its associated *adjacency list* and *adjacency matrix*. In addition, state, with justification, which of these two methods would be the "better" way to represent the graph. **[10 marks]**

[TOTAL 10]

Section B

Answer THREE questions from this section.

Question 5

1. Binary Search

   Consider the following array:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
value	11	13	17	23	29	31	37	41	43	47	53	59	61	67	71

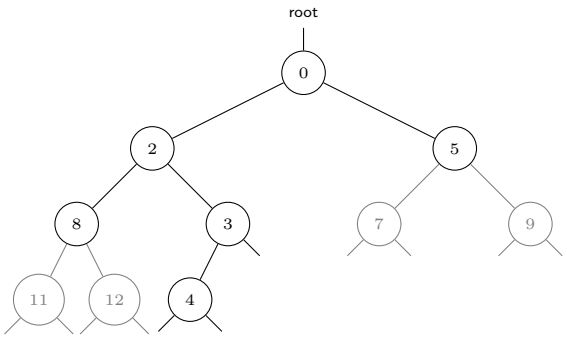
   Suppose you try to find the value 45 in this array using binary search. For each iteration of the algorithm, give the values of the `first`, `middle`, and `last` indices, and a (short) explanation of the changes. **[10 marks]**

2. Searching in a heap

   Unlike binary search trees, heaps do not lend themselves to binary search. We can still try to improve on blind exhaustive search, though.

   Each node in a heap holds the smallest value in the subtree below it. Therefore, if the value in a node is bigger than what we are searching for, we can ignore the entire subtree below that node.

   For example, while searching the value 4 in this min-heap:



   when the search encounters the values 5 and 8, it knows that all values below those nodes will be too large, so it can ignore the gray parts of the heap.

   Write a search function which uses this idea. It should return the index where the value was found (or -1, if it is not in the heap). Complete either the Java or C++ version below. You will probably need a recursive auxiliary function.

- Java version:

```
public class HeapSearch{
 public static int search(int[] heap, int findMe){
  // TO DO
 }
}
```

- C++ version:

```
int heapSearch(vector<int> &heap, int findMe){
 // TO DO
}
```

[10 marks]

[TOTAL 20]

Question 6

1. Consider the following array:

index	0	1	2	3	4	5	6	7
value	6	4	8	2	5	3	7	1

   We want to sort this array using Bubble Sort. Write down the contents of the array after each iteration of the main loop. **[10 marks]**

2. Come up with an algorithm for sorting arrays (or C++ vectors) of **boolean** values.

   This specialised algorithm should have time complexity $O(n)$. Note that this is better than what is possible in the general case; think about how you can use the limited data domain (only two different values!) to achieve this.

   - Java version:

```
public class BitSort{
 public static void sort(boolean[] values){
  // TO DO
 }
}
```

   - C++ version:

```
void bitSort(vector<bool> &values){
 // TO DO
}
```

[10 marks]

[TOTAL 20]

Question 7

(a) Define what is meant by a *Binary Search Tree* (BST). Give an algorithm in pseudo code to *insert* a value into a BST.

Using your insertion pseudo code create a BST by inserting the following list of integers:

$$17,\ 21,\ 10,\ 12,\ 2,\ 61,\ 50,\ 78$$

one number at a time, in the order given, into an initially empty BST. Illustrate the result of each insertion by drawing the BST after each insertion, i.e. 8 BST diagrams in total. **[10 marks]**

(b) Give a pseudo code algorithm to *delete* a specified value from a BST, such that the resultant tree is still a BST.

Illustrate your answer by stating the steps involved to remove the number 17 from the BST you have constructed in Part **(a)**, and by drawing the resultant BST. **[10 marks]**

[TOTAL 20]

Question 8

You are given the following graph:

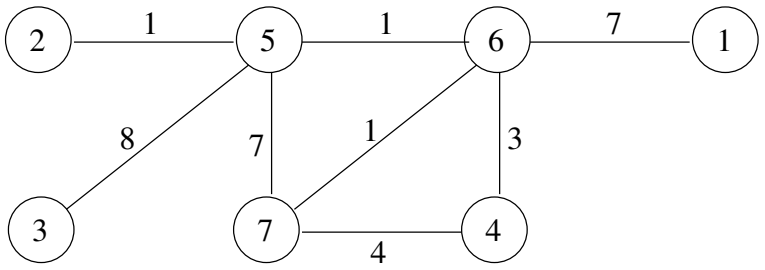


Figure 1: A 7 node graph

(a) Give a brief overview of E.W. Dijkstra's Single Source Shortest Path algorithm. This should include its purpose, input, output, approach and complexity. **[6 marks]**

(b) Apply Dijkstra's *Single Source Shortest Path* algorithm to the graph given in Figure 1. Use *node 1* as the source. Give the algorithm's output for the graph, including the paths. **[14 marks]**

[TOTAL 20]

END OF EXAMINATION PAPER