Semester two of academic year (2015—2016) of BJUT

Data Structures and Algorithms I

COMP2002J

Exam Instructions: Answer 2 of 3 parts

Honesty Pledge:

I have read and clearly understand the Examination Rules of Beijing University of Technology and University College Dublin and am aware of the Punishment for Violating the Rules of Beijing University of Technology and University College Dublin. I hereby promise to abide by the relevant rules and regulations by not giving or receiving any help during the exam. If caught violating the rules, I would accept the punishment thereof.

Pledger:_____ Class No:_____
BJUT Student ID:_____ UCD Student ID:_____

Notes:

The exam paper has 3 parts on 6 pages, with a full score of 100 points. You are required to use the given Examination Book only.

Instructions for Invigilators:

Candidates are allowed to use non-programmable calculators during this examination.

Obtained Score

Part: 1 Stack and Complexity

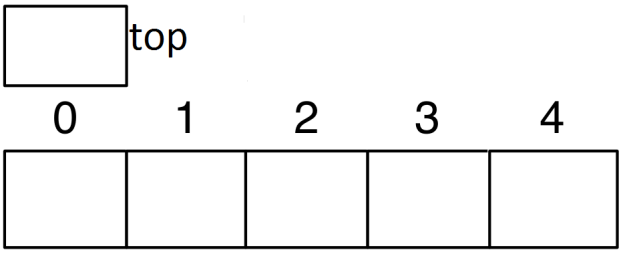
a. Give the definition of the stack abstract data type. List the operations provided, and the complexity of each operation in big O notation.

(6 Points)

b. Analyse the following operations performed on a newly created empty stack. Every time an item is popped, this item is printed. Give the output of these operations.
`push(45), push(23), push(49), pop(), pop(), push(27), push(99), pop()`

(4 Points)

c. Copy the drawing below in your answer book and show the contents of the variable `top` (which starts at 0) and the array `values` in the array based implementation of the Stack abstract data type after every third operation has been completed, there should be 3 diagrams in total.
`push(15), push(23), pop(), push(99), push(45), pop(), push(67), push(66), pop()`



(5 Points)

d. Based on the code below, write the code for the push method in the array implementation of the stack abstract data type.

```
public class ArrayStack implements Stack {
  private int top;
  private int[] values;
  ...
}
```

(8 Points)

e. Determine the complexity of the following piece of code in big O notation? Explain your answer.

```
public int answer(int c) {
  int sum = 0;
```

```
    for (int i = 0; i < c; i++) {
      sum = sum - c;
    }
    return c;
  }
```

(5 Points)

f. Determine the complexity of the following piece of code in big O notation? Explain your answer.

```
public int answer(int b) {
  int sum = 0;
  for (int i = 0; i < b; ++i) {
    return b;
  }
}
```

(5 Points)

g. Determine the complexity of the following piece of code in big O notation? Explain your answer.

```
public int answer(int a) {
  int sum = 0;
  for(int i = a; i > 0; i /= 2){
    sum += i;
  }
  return sum;
}
```

(5 Points)

h. Assume that you are working with a programming language where the only data structure available is the stack. Explain **in detail** how you would use multiple stacks to implement the enqueue and dequeue operations of a queue and maintain the correct order of output. Illustrate your explanation with a diagram

(12 Points)

(Total 50 Points)

Obtained Score

Part: 2 Lists and Sorting

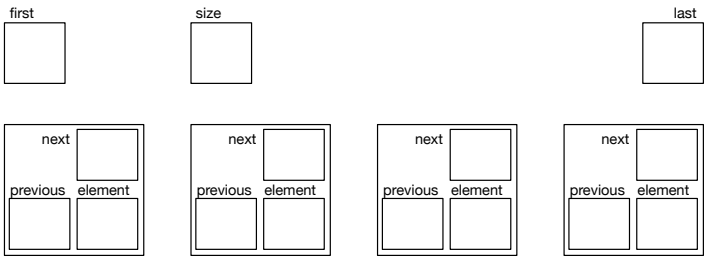
a. We have studied 3 implementations of the List abstract data type, which of the three is the most efficient? Explain your answer by comparing the complexity of some operations in the three implementations.

(5 Points)

b. Describe in English the steps involved in searching a doubly linked list to check if it contains an element X. What is the complexity of the operation in big O notation

(5 Points)

c. Draw a diagram similar to the one below. This diagram should show the state of a doubly linked list after the following operations have been completed.
The references should be drawn as arrows and the values should be shown
`addFirst(A), addLast(B), p = first(), addAfter(p, C), l = last(), addAfter(l,D), p = after(p), addBefore(p, E)`



(5 Points)

d. Based on the code below, write the code for the `insertFirst` method for the doubly linked list implementation. This method should take a single parameter of an object to be added to the list. Remember that the method should return a `Position`.

```
public class DoublyLinkedList implements List{
  private Node first;
  private Node last;
  private int size;
  ...
```

}

(10 Points)

e. What is the complexity of the insertion sort algorithm in big O notation?

(2 Points)

f. Explain in your own words how the quicksort algorithm functions. What is the complexity of this algorithm in big O notation? If the pivot value is chosen as the first number in the partition, how can different types of data change the running time of the algorithm?

(12 Points)

g. Apply the mergesort algorithm to the following array {2, 78, 7, 45, 9, 33, 99, 1}. Draw a diagram showing the partitions of the array after each application of the algorithm. Show the parameters of the calls to the mergesort method each time it is called.

(3 Points)

h. Describe the operation of radix sort. Illustrate this explanation with an example. Show the order of the array after each pass while sorting the following array 19, 29, 34, 78, 67, 58. The algorithm should use the radix 10.

(8 Points)

(Total 50 Points)

Obtained Score

Part: 3 Maps

a. What is a Map? List the and describe the operations of the Map ADT.

(10 Points)

b. Hash functions convert keys to integer values in the correct range. Describe the two basic mappings?

(4 Points)

c. The polynomial sum is a method to convert a sequence into a hash value. Describe how this process works. Calculate the hash code for

the following string "abba" if 'a' = 97, 'b' = 98 and the polynomial used is 3.

(6 Points)

d. A hash table implementation for storing entries whose keys are integer values consists of an array of size 11. It uses the hash function $h(x) = (4x + 7) mod 11$. The collision strategy is linear probing. Show using a diagram, the state of the hash table after each of the following keys has been added. {45, 33, 23, 78, 29, 13, 54, 61}

(10 Points)

e. Based on the code below, write the `get` method for the array based implementation of the map using separate chaining as a collision strategy. You should also fully explain the separate chaining strategy for collision handling and comment on how it is implemented in the code.

```
public class HashMap implments Map{
  private List<Entry>[] entries;
  private int size;
  ...
  public int hashCode(int k){
    return k % entries.length;
  }
}
```

(20 Points)

(Total 50 Points)