NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

OLLSCOIL NA hÉIREANN MÁ NUAD

THE NATIONAL UNIVERSITY OF IRELAND MAYNOOTH

JANUARY 2009 EXAMINATION

CS210

Algorithms & Data Structures 1

Dr. P. Morrow, Dr. A. Winstanley, Dr. P. Maguire

Time allowed: 2 hours

Answer ***three*** questions

All questions carry equal marks

[25 marks]

1 (a) Provide a brief description of the following features of object-oriented programming languages. Mention how they facilitate efficient programming. **[6 marks]**

i. Encapsulation
ii. Inheritance

(b) Explain with the aid of a diagram what the Big-O complexity of an algorithm means. Why is it important to know the complexity of an algorithm? **[5 marks]**

(c) Consider a problem where every person in a room has to shake hands with everybody else. Derive a function *f(n)* which describes how the number of handshakes increases with respect to the number of people in the room, *n*. **[5 marks]**

(d) A function *f(n)* describes the running time of an algorithm where *n* is the size of the problem: **[5 marks]**

$$f(n) = \frac{n\log n - n}{3} + 7$$

State the Big-O complexity of the algorithm and prove that this is the case using the mathematical definition.

(e) State the Big-O complexity of the following code and explain your reasoning. **[4 marks]**

```
for(int i = n; i < n+10; i++){
     for(int j = n; j < 10; j++){
         array[j]+=i;
}
```

[25 marks]

2 (a) Complete the following Java method for returning the maximum value in an array of ints. Describe how the algorithm works. **[5 marks]**

```
public int findMax(){

     …fill this in…
     return max;
}
```

(b) Describe the difference between linear search and binary search using suitable examples. State the Big O Complexity of these algorithms. **[6 marks]**

(c) Describe in your own words an algorithm for inserting elements **[5 marks]** into an ordered array. Provide examples and diagrams as appropriate.

(d) Complete the following Java method for swapping two Strings in an array called `myArray`. The input parameters denote the array indices of the two Strings to be swapped. **[4 marks]**

```
public void swap(int one, int two){

     …fill this in…
}
```

(e) Compare the advantages and disadvantages of using an array and a linked list to store data. Describe the applications to which each data structure is suited. **[5 marks]**

[25 marks]

3 (a) Explain the bubblesort algorithm in your own words and provide a Java implementation. **[5 marks]**

(b) What are the best case and worst case running times for insertion sort and when do they arise?
Sort the following numbers using insertion sort, outlining each of the steps involved: **[5 marks]**

```
[ 9 4 8 2 6 3 ]
```

(c) With the aid of suitable diagrams and examples, describe how a queue can be implemented using an array. **[6 marks]**

(d) What are the contents of an initially empty stack following each of these operations? **[3 marks]**

```
push (5)
push (6)
pop ()
peek ()
push (2)
pop ()
```

(e) Provide a full Java implementation of a Stack class. The stack should be implemented using an array of ints. You will need a variable to track the top of the stack. The constructor should take in the size of the stack as a parameter and initialize the array. The following methods should be included in the class: **[6 marks]**

i. push
ii. pop
iii. isEmpty
iv. isFull

[25 marks]

4 (a) Outline the defining characteristics of a recursive algorithm. **[3 marks]**

(b) Describe the recursive mergesort algorithm in your own words and show how the following numbers are sorted using mergesort: **[7 marks]**

```
[ 2 4 8 6 5 7 3 1 ]
```

(c) Explain the concept of a linked list in your own words. Provide examples and diagrams as appropriate. **[5 marks]**

(d) Show, using appropriate diagrams, the steps involved in inserting an element into the middle of a doubly-linked list. **[5 marks]**

(e) Explain in detail what the following code does and how it does it: **[5 mark]**

```
public void unknown(long data){
   Link newLink = new Link(data);
   if( isEmpty() ){
       last = newLink;
       first.previous = newLink;
   }
   newLink.next = first;
   first = newLink;
}
```