UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

Department of Computing

B. Sc. Examination 2014

IS52017C
Algorithms

Duration: 2 hours 15 minutes

Date and time:

There are five questions in this paper. You should answer no more than THREE questions. Full marks will be awarded for complete answers to a total of THREE questions. Each question carries 25 marks. The marks for each part of a question are indicated at the end of the part in [.] *brackets.*

There are 75 marks available on this paper.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1

(a) i. Define formally what it means for an array a of N integers to be sorted in ascending order.

ii. In order to sort a list of objects of type T what property must the elements of type T have?

iii. Give two different orderings on Strings and say how your two definitions would cause the array of Strings `{"dogs","cat","person"}` to be sorted in ascending order?

[9]

(b) Describe how the *merge sort* algorithm works. When doing this, do not give the algorithm for merging but informally specify what the *merge* algorithm should do and give an example. [8]

(c) Here is a method for merging two sorted lists used in Merge Sort:

```
static int [] merge( int[] a, int [] b)
{
 int N=a.length+b.length;
 int [] c = new int[N];
 int i=0,j=0,k=0;
 while (i<a.length && j <b.length)
 {
   if (a[i] < b[j]) {c[k]=a[i];i++;}
   else {c[k]=b[j];j++;}
   k++;
 }
 if (i==a.length) for(int z=j;z<b.length;z++) c[a.length+ z] =b[z];
 else for(int z=i;z<a.length;z++) c[b.length+ z] =a[z];
 return c;
}
```

What would happen is we applied it to two non-sorted arrays? For example if `c={1,3,2}` and `d={2,1,5}`. What would `merge(c,d)` return? [8]

Question 2

(a) Here are the list axioms for list$[T]$:

$$\text{nil} \in \text{list}[T]$$
$$\text{cons} : T \times \text{list}[T] \to \text{list}[T]$$
$$\text{head} : \text{list}[T] \to T$$
$$\text{tail} : \text{list}[T] \to T$$

i. head(nil) $=$ error
ii. tail(nil) $=$ error
iii. head(cons$(x, m)) = x$
iv. tail(cons$(x, m)) = m$

Prove using the axioms that

$$\text{head}(\text{tail}(\text{cons}(1, \text{cons}(2, \text{nil})) = 2.$$

[9]

(b) i. Define length : list$[T] \to \mathbb{N}$
ii. Prove using the List Axioms and the definition of length that

$$\text{length}(\text{tail}(\text{cons}(1, \text{cons}(2, \text{nil})) = 2.$$

[8]

(c) Here is a Java implementation of list$[T]$.

```
import java.util.*;
public class genericLists  <T>
{
 public  T head (ArrayList  <T> m)
 {
  return m.get(0);
 }

 public  ArrayList <T>  tail (ArrayList <T>  t)
 {
  ArrayList <T> m= new ArrayList <T> (t);
  m.remove(0);
  return m;
 }

 public  ArrayList <T>  nil ()
 {
   return new ArrayList <T>();
 }

 public  ArrayList  <T> cons (T t, ArrayList <T> m)
 {
     ArrayList <T> k= new ArrayList <T>();
     k.add(t);
     k.addAll(m);
     return k;
 }


}
```

The append function satisfies the following rules.

$$\text{append} : \text{list}[T] \times \text{list}[T] \rightarrow \text{list}[T]$$

$$\text{append}(\text{nil}, m) = m$$

$$\text{append}(\text{cons}(x, k), m) = \text{cons}(x, \text{append}(k, m))$$

Write a recursive Java method that implements append. [8]

Question 3

(a) Let $x = 2^y$ Which of the following is true:

i. $y = log_e(x)$

ii. $y = log_2(x)$

iii. $x = log_2(y)$

iv. None of the above.

[2]

(b) Which of the following functions will produce the largest values (asymptotically) as N gets bigger:

i. $f(N) = 30 * N$

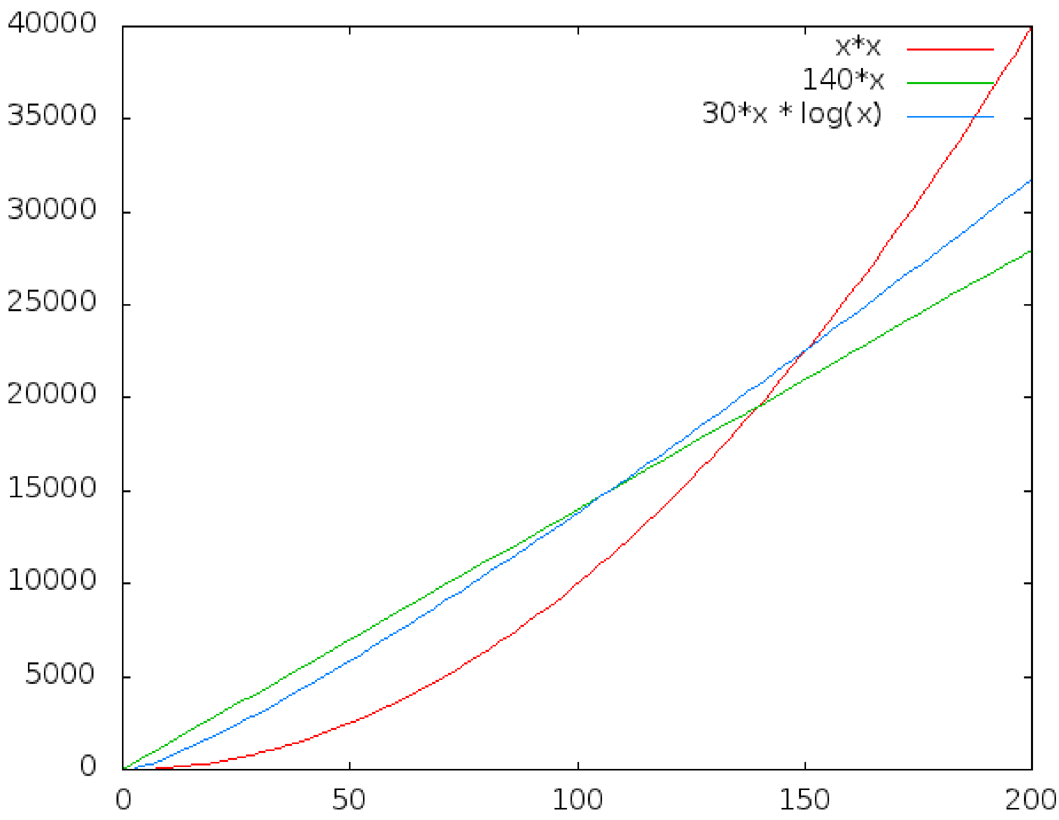
ii. $f(N) = N^2$

iii. $f(N) = N * log(N)$

iv. $f(N) = 10000000 * N$.

[2]

(c) Consider the following plot:



What is the equation of the middle curve in the plot. i.e. the one which has the value of just over 30000 when x has the value 200?

i. $x * x$
ii. $140 * x$
iii. $30 * x * log(x)$
iv. None of the above.

[2]

(d) The time complexity of *insertion sort* is

i. $O(N)$
ii. $O(N^2)$
iii. $O(N * log(N))$
iv. None of the above.

[2]

(e) The time complexity of *merge sort* is

i. $O(N)$
ii. $O(N^2)$
iii. $O(N * log(N))$
iv. None of the above.

[2]

(f) If it take 5 nanoseconds to sort 10 elements using insertion sort, roughly how long will it take to sort 20 elements?

i. 10 nanoseconds
ii. 20 nanoseconds
iii. 30 nanoseconds
iv. None of the above.

[2]

(g) What is the time-complexity of this function in terms of N?

```
int f(int N)
{
  int total=0;
  for (int i=0;i<N;i++)
    for (int j=0;j<N;j++)
```

```
      total=total+i+j;
  return total;
}
```

i. linear
ii. quadratic
iii. exponential
iv. None of the above.

[2]

(h) What is the time-complexity of this function in terms of N?

```
int f(int N)
{
  if (N<2) return 1;

  return f(N-1)+f(N-2);
}
```

i. linear
ii. quadratic
iii. exponential
iv. None of the above.

[2]

(i) Here are two methods for computing x^n:

```
static int powerA(int x, int n)
{
     int total=0;
     while (n>0) {total=total*x;n--;}
     return total;
}


static int powerB(int x, int n)
{
     if (n==0) return 1;
     int k=n/2;
     int z=powerB(x,k);
     int r=z*z;
     if (n%2==0) return r;
```

```
    return x*r;
}
```

Which one of the following is true

i. `powerA` is linear and `powerB` has quadratic time-complexity.
ii. `powerA` is exponential and `powerB` has log(N) time-complexity.
iii. `powerA` is quadratic and `powerB` has exponential time-complexity.
iv. None of the above.

[2]

(j) Here are two methods for computing x^n:

```
static int powerA(int x, int n)
{
     int total=0;
     while (n>0) {total=total*x;n--;}
     return total;
}


static int powerB(int x, int n)
{
     if (n==0) return 1;
     int k=n/2;
     int z=powerB(x,k);
     int r=z*z;
     if (n%2==0) return r;
     return x*r;

}
```

Which one of the following is true

i. `powerA` is more efficient than `powerB`.
ii. `powerB` is more efficient than `powerA`.
iii. `powerB` has the same efficiency as `powerA`.
iv. One of the methods has an error.

[2]

(k) Write a method whose heading is

```
HashMap <String,Integer> occurrences (ArrayList <String> x)
```

which returns a HashMap, mapping each String s in `x` to the number of occurrences of s in `x`. [5]

Question 4

(a) i. What is a *spanning tree* of a Graph. Give an example.

ii. What is the purpose of Dijkstra's Algorithm. Give an example where it could be used in practice.

iii. What is the purpose of Prim's Algorithm. Give an example where it could be used in practice.

[9]

(b) Prove that if $v_1 \to v_2 \to w_1 \to \ldots \to w_m \to v_n$ is a shortest path between v_1 and v_n in a graph G then $v_2 \to w_1 \to \ldots \to w_m \to v_n$ is a shortest path between v_2 and v_n in G. [8]

(c) Given the abstract class `abstractGraph` below, for undirected graphs whose vertices are of type T, write a method, which calls the abstract methods in `abstractGraph` which returns the set of all isolated vertices in the graph. An isolated vertex is one with no neighbours.

```
public abstract class abstractGraph <T>
{
  public abstract Set <T> neighbours(T v); // the set of neighbours of vertex v
  public abstract Set <T> vertices(); // the set of all vertices in the graph
}
```

[8]

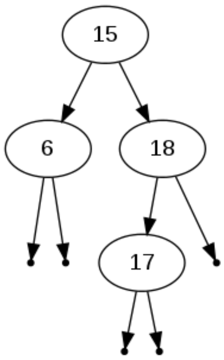
Question 5

(a) A Binary Tree whose nodes of type T can be defined as follows:

$$\textit{empty} \in \textit{BT}[T]$$

$$\textit{consBT} : T \times \textit{BT}[T] \times \textit{BT}[T] \to \textit{BT}[T]$$

i. Construct the following tree using the above functions *empty* and *consBT*

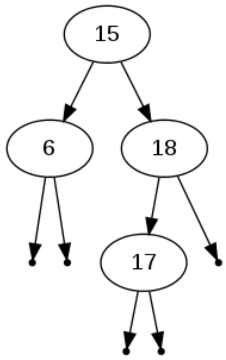


above. [3]

ii. Consider the following Java classes:

```
public abstract class binaryTree <T>
{
}


class emptyTree  <T> extends binaryTree  <T>
{
}



class consbinaryTree <T> extends binaryTree <T>
{
    T root;
    binaryTree <T> left;
    binaryTree <T> right;

    consbinaryTree (T roo, binaryTree <T> l, binaryTree <T> r)
    {root=roo;left=l;right=r;}
}
```

What is the Java expression that generates the binary tree:



[3]

(b) The *depth* function on binary trees is defined as follows:

$$\textit{depth} : \textit{BT}[T] \to \mathbb{N}$$

$$\textit{depth}(\textit{empty}) = 0$$
$$\textit{depth}(\textit{consBT}(x, b_1, b_2)) = 1 + \textit{max}(\textit{depth}(b_1), \textit{depth}(b_2))$$

i. Define a *depth* method for the class `binaryTree <T>`.
ii. Define a *depth* method for the class `emptyTree <T>`.
iii. Define a *depth* method for the class `consbinaryTree <T>`. You may assume the existence of a $\max$ method.

[6]

(c) The functions *left* and *right* on *BT*$[T]$ are of the following types:

$$\textit{root} : \textit{BT}[T] \to T$$
$$\textit{left} : \textit{BT}[T] \to \textit{BT}[T]$$
$$\textit{right} : \textit{BT}[T] \to \textit{BT}[T]$$

and satisfy the following axioms:

$$\textit{root}(\textit{consBT}(x, b_1, b_2)) = x$$
$$\textit{left}(\textit{consBT}(x, b_1, b_2)) = b_1$$
$$\textit{right}(\textit{consBT}(x, b_1, b_2)) = b_2$$

i. Define methods *left* and *right* for the class `binaryTree <T>`.
ii. Define methods *left* and *right* for the class `consbinaryTree <T>`.

[4]

(d) Write a method whose heading is
`static Integer least(BT <Integer> b)`
which returns the smallest element of a non-empty Binary Search Tree b of Integers. (Assume the existence of an `isEmpty()` method on binary trees.) [9]