THIS PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION HALL

UNIVERSITY OF LONDON

CM2035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures 2

MOCK EXAM

Date 2020: time

Time allowed: 2 hours

DO NOT TURN OVER UNTIL TOLD TO BEGIN

INSTRUCTIONS TO CANDIDATES:

This examination paper is in two parts: Part A and Part B. You should answer **ALL** of question 1 in Part A and **TWO** questions from Part B. Part A carries 40 marks, and each question from Part B carries 30 marks. If you answer more than **TWO** questions from **Part B** only your first **TWO** answers will be marked.

All answers must be written in the answer books; answers written on the question paper will not be marked. You may write notes in your answer book. Any notes or additional answers in the answer book(s) should be crossed out.
The marks for each part of a question are indicated at the end of the part in [.] brackets. There are 100 marks available on this paper.

Calculators are not permitted in this examination.

MOCK EXAM, PART A

Candidates should answer **ALL** of Question 1 in Part A.

Question 1

(a) Algorithm F1 returns the amount of ‘valley’ numbers found in a square matrix made of N rows and N columns. A ‘valley’ number is a number whose left and right immediate neighbours store values greater than the value stored in the ‘valley’. For example, in the following matrix there is only 1 ‘valley’ number (number 3 in position (1,1)). Elements in the left and right borders of the matrix cannot be ‘valley’ numbers (as they lack one neighbour).

$$\begin{array}{c|c|c|c|}
 & [0] & [1] & [2] \\ \hline
[0] & 2 & 7 & 4 \\ \hline
[1] & 5 & 3 & 9 \\ \hline
[2] & 8 & 6 & 1 \\ \hline
\end{array}$$

```
M: matrix of integer numbers
N: number of rows (columns) of M

function F1(M,N)
    valley=0
    for 0 <= i < N
        for 1 <= j < N-1
            if(M[i,j]<M[i,j-1] AND M[i,j]<M[i,j+1])
                valley=valley+1
    return valley
end function
```

Select ALL statements that apply. [4]

i. The worst and best case time complexity of F1 are the same
ii. The worst-case time complexity of F1 is $\Theta(N^2)$
iii. The best-case time complexity of F1 is $\Theta(N)$
iv. The best-case time complexity of F1 is $\Theta(1)$
v. The worst-case time complexity of F1 is $\Theta(N*(N-2))$.

Correct answers: Only i and ii

(b) Consider the following recursive algorithm:

```
A: array of integer numbers
N: number of elements in A
x: integer number

function F2(A,N,x):
    if(N==0):
        return 0
    if(A[N-1]==x):
        return 1+ F2(A,N-1,x)
    return F2(A,N-1,x)
end function
```

Assume that the array A is equal to [5, 4, 3, 2, 1, 2, 3, 4, 5]. What is the value returned by F2(A,9,3)? [4]

Correct answer: 2

(c) Consider the same recursive algorithm of part (b):

```
A: array of integer numbers
N: number of elements in A
x: integer number

function F2(A,N,x):
    if(N==0):
        return 0
    if(A[N-1]==x):
        return 1+ F2(A,N-1,x)
    return F2(A,N-1,x)
end function
```

What is the recurrence equation describing its worst-case time complexity? [4]

Choose ONE option:

i. $T(N)= T(N-1) + N$
ii. $T(N)= T(N-1) + C$ (C is a constant)
iii. $T(N)= T(N/2) + C$ (C is a constant)
iv. $T(N)= T(N/2) + N$
v. None of the others

Correct answer: ii

(d) What pseudocode fragment should replace Z in the following comparison-based sorting algorithm?

```
A: array of integer numbers
N: number of elements in A

function Sort(A,N)
   for 1 <= j <= N-1
      ins=A[j]
      i=j-1
      while (i>=0 and ins<A[i])
            Z
           i=i-1
      end while
      A[i+1]=ins
      end for
end function
```

Choose ONE option:

i. A[i]=A[i+1]
ii. A[i+1]=A[i]
iii. ins=A[i]
iv. A[j]=ins
v. None of the others

Correct answer: ii (`A[i+1]=A[i]`)

(e) What of the following statements are **false**?

Select ALL statements that apply. [4]

i. The worst-case time complexity of a comparison sort cannot be better than $\Theta(N\log N)$
ii. Best-case time complexity of comparison sorts can be $\Theta(N)$
iii. Radix sort cannot be used to sort decimal numbers
iv. Mergesort is one of the comparison sorts with best worst-case performance
v. The best-case time complexity of non-comparison sorts is $\Theta(1)$

Correct answer: iii, v

(f) An 5-element hash table uses linear probing to deal with collisions. The hash function is h(k)=(2*k+1)%5. Assume the hash table starts empty. What is the content of it after inserting the following numbers (in this order): 4, 27, 10, 9, 12? [4]

Choose ONE option:

i. [27, 10, 9,12, 4]
ii. [27, 10, -1, -1, 4]
iii. [27->12, 10, -1,-1, 4->9]
iv. [4, 9, 10, 27, 12]
v. None of the others

Correct answer: i

(g) Consider the following linked list:

head/0xA → 7/0xD → 3/NULL

A new node is inserted between nodes storing numbers 7 and 3. Assume the new node stores number 10 and it is allocated memory address 0xE. What are the contents of the new list? [4]

Choose ONE option:

i. head/0xA → 7/0xD →10/0xE → 3/NULL
ii. head/0xE → 7/0xA → 10/0xD → 3/NULL
iii. head/0xA → 7/0xE → 10/0xD → 3/NULL
iv. head/0xA → 7/0xD → 10/NULL → 3/0xE
v. None of the others

Correct answer: iii

(g) The following numbers are inserted in a Binary Search Tree (in this order): 5,1,6,12,13,22

What information in printed on screen when traversing the tree with the algorithm shown below? [4]

```
function traverse(T)
      if(T.root!=NULL)
            traverse(T.right)
            traverse(T.left)
            print(T.root)
end function
```

Choose ONE option:

i. 1, 5, 6, 12, 13, 22
ii. 1, 22, 13, 12, 6, 5
iii. 6, 12, 13, 22, 1, 5
iv. 22, 13, 12, 6, 1, 5
v. None of the others

Correct answer: iv

(h) The following array: [8, 3, 7, 2, 1, 9, 4] is transformed in a min-heap in place.

What is the content of the array storing the min-heap? Position 0 in the array is the leftmost position [4]

Choose ONE option:

i. [1, 3, 7, 2, 8, 9, 4]
ii. [1, 2, 3, 4, 7, 8, 9]
iii. [1, 2, 4, 8, 3, 9, 7]
iv. [9, 3, 8, 2, 1, 7, 4]
v. None of the others

Correct answer: iii

(h) Consider the graph represented by the following adjacency matrix:

$$\begin{array}{c|ccccc|}
 & A & B & C & D & E \\
\hline
A & 1 & 8 & 12 & 3 & 1 \\
B & 8 & 9 & 5 & 6 & 4 \\
C & 12 & 5 & 11 & 13 & 7 \\
D & 3 & 6 & 13 & 4 & 2 \\
E & 1 & 4 & 7 & 2 & 15 \\
\hline
\end{array}$$

What is the cost of the minimum spanning tree? [4]

Choose ONE option:

i. 9
ii. 11
iii. 8
iv. 10
v. None of the others

Correct answer: v (correct answer: 12)

MOCK EXAM PART B

Candidates should answer any **TWO** questions from Part B.

Question 2

In an augmented binary search tree (BST) every node is augmented with the field *size(x)* equal to the number of nodes in the sub-tree rooted in x. Figure 1 below shows an example of an augmented BST.

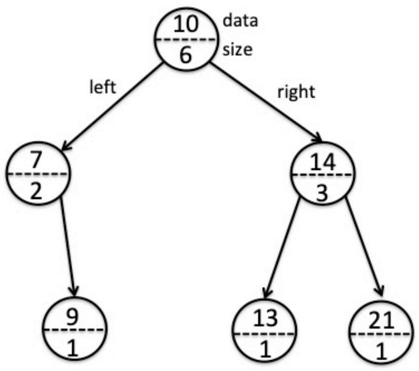

Figure 1. An augmented BST

Each node in an augmented BST has 4 parts: the data part (data), the size part (size), the memory address of the left child (left) and the memory address of the right child (right).

The following algorithms, A1 and A2, solve the same problem. To do so, they receive as input arguments:

root: the root of an **augmented binary search tree** storing integer numbers
x: an integer number

ALGORITHM 1

```
function A1(root, x)
   Q = new Queue()
   ENQUEUE(Q,root)
   while !ISEMPTY(Q)  do
        t = PEEK(Q)
        if (t->data>=x)
             return t->size
        else
        ENQUEUE(Q,t->left)
        ENQUEUE(Q,t->right)
        DEQUEUE(Q)
   end while
   return 0
end function
```

Note: the function ENQUEUE only inserts a new element in the queue if this element is different from NULL.

ALGORITHM 2

```
function A2(root,x)
   if (root==NULL)
      return 0
   else
      if(root->data >= x)
          return root->size
      else
          return A2(root->right,x)
end function
```

(a) For the augmented BST shown in Figure 1, what is the content of the queue **immediately before returning** from the execution of A1(root,14)? Assume that ENQUEUE(node) stores the pair (data, size) in the queue. [4]

Answer: {(14,3) ,(9,1)}

(b) For the augmented BST shown in Figure 1, what is the return value of A2(root, 21)? [4]

Answer: 1

(c) What is the task performed by algorithms A1 and A2? [4]

Answer: Returning the number of nodes in the BST whose data field is equal to or greater than the value of x.

(e) What is the worst-case time complexity of A1? Use Theta notation [1] and explain your reasoning [3] [4]

Answer: Theta(N). In the worst-case the while loop is executed N times

(d) Assuming a fully balanced BST (a fully balanced BST has all its levels fully populated) of N elements, what is the recurrence equation describing the worst-case running time of A2? [4]

Answer: T(N)= T(N/2)+C

(f) What is the worst-case time complexity of A2? Use Theta notation [1] and show your workings [3] [4]

Answer: Theta(logN) [1]
Applying the Master Theorem:

- a=1; b=2; logba=0 => Case 2 => Theta(logN)

Solving the recurrence equation by expanding it

- T(N)=T(N/2)+C
-     =T(N/4)+2C
-     …
-     =T(N/2^k)+k*C
-     =T(1)+logN*C, as T(1) takes constant time => T(N) is Theta(logN)

(g) Which algorithm one do you recommend to implement to solve the problem? Why? [6]

Answer
The student can recommend any algorithm, as long as there is a reasonable justification for it.

Case 1: Student recommends A1. In this case the student must mention that, although the worst-case time complexity of A1 is higher than of A2 (Theta(N) vs Theta(logN)), it would be a good choice if the size of the tree makes the recursion depth too high for a computer to deal with it. That is, A1 is good for cases where the recursive implementation is unfeasible.

Case 2: Student recommends A2. In this case the student must mention that A2 has a lower worst-case time complexity than A1, which makes it faster for large values of N.

Question 3

The data structure shown in Figure 2 can be thought as a list of playlists. Every element in the main list (shadowed nodes) stores the name of the music genre stored in its corresponding playlist. For the sake of simplicity, in this question every song in a playlist is identified by a number (instead of a name).

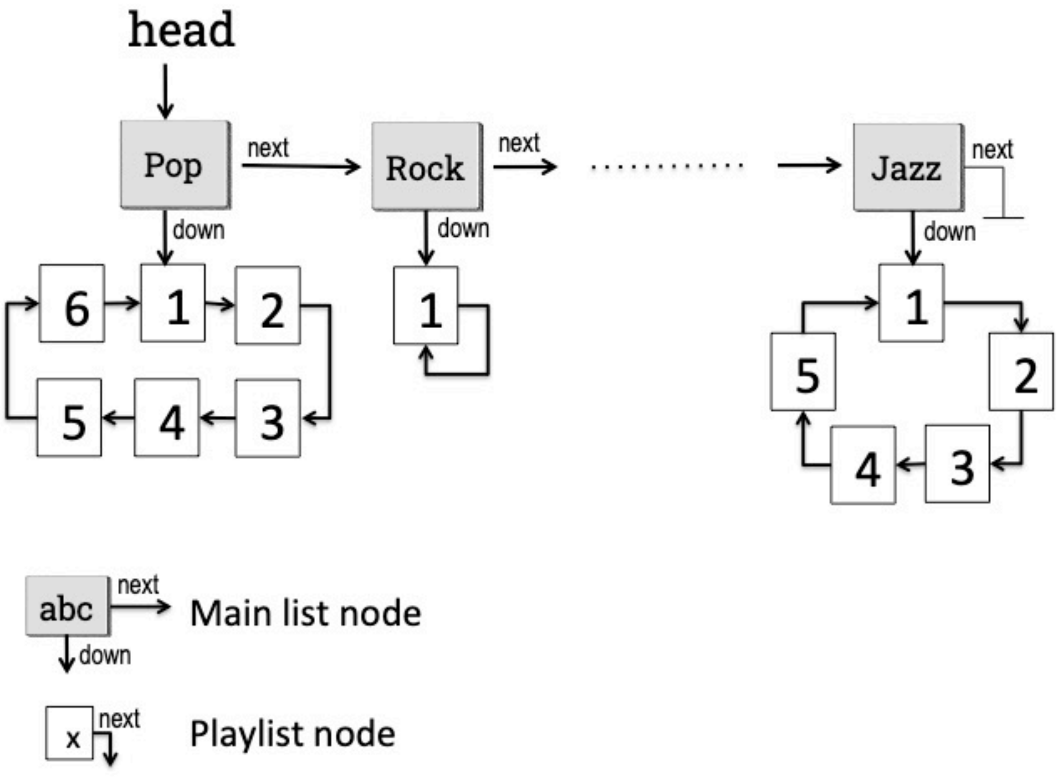

Figure 2. A list of playlists

(a) In a single linked list implemented in C++ this is the definition of a node: [4]

```
struct Node {
	int data;
	Node *next;
  };
```

This definition is useful for the playlist nodes, but not for the nodes belonging to the main list. The nodes in the main list need two pointers (one for the next node in the main list and one for the first node in the playlist). Assuming that the above definition is kept for the nodes of the playlist, propose a new definition of node for the nodes of the main list. Call this type of node Node_main and use the names of pointers given in Figure 2.

Answer: (assign 1.5 points per correct line)

```
struct Node_main {
        char *data;  //they could also use an array of char
        Node_main *next;
        Node *down;
    };
```

(b) Write the pseudocode of the function SEARCH_MUSIC_GENRE(head, genre) that receives as input arguments the head of the list of playlists and the name of a music genre (e.g. Pop, Rock, Blues, etc). If a node with that music genre exists in the main list, the function SEARCH_MUSIC_GENRE(head, s) must return the memory address of the first element of the corresponding playlist. If a node with that music genre does not exist in the main list, the value NULL must be returned. [6]

ANSWER

function SEARCH_MUSIC_GENRE(head, genre)

```
    //checking empty main list  [0.5]
    if(head==NULL)
          print("Empty main list")
          return
    //searching for the node with music genre 'genre' in main list
     tmp=head  //initialise traversing pointer [1]
     while(tmp!=NULL)  //loop to find genre [1]
          if(tmp->data == genre) //condition to stop loop [1]
             return tmp->down   //return correct pointer [1]
          tmp=tmp->next  //move to next element [1]
     return tmp  //return value [0.5]
end function
```

(c) Write the pseudocode of the function INSERT_SONG(head, genre) that inserts a new number in the playlist corresponding to the music genre ‘genre’. For the example given in Figure 2, INSERT_SONG(head, “Pop”) will insert a node storing number 7 between nodes 6 and 1. Assume that, if the node with the music genre ‘genre’ already exists in the main list, then the corresponding playlist has at least one element. If the node with the music genre ‘genre’ does not exist, you must create it and insert the first song. You can use the function SEARCH_MUSIC_GENRE() in this pseudocode (even if you did not write its pseudocode in the previous question). [10]

ANSWER

```
function INSERT_SONG(head,genre)
   //checking empty main list [1]
   if(head==NULL)
         print(“Empty main list”)
         return
    //getting the pointer to the correct playlist [1]
   tmp=SEARCH_MUSIC_TYPE(head,genre)

  //case where a new main list node must be created with one song [4]
   if(tmp==NULL)
      tmp=head
      while(tmp->next!=NULL)
             tmp=tmp->next
      //new main list node
      Node_main newNode_music=new Node_main
      newNode_music->data=genre
      newNode_music->next=NULL
      newNode_music->down=NULL
      //connect new main list node to the list
      tmp->next=newNode:music

      //new playlist node
      Node newNode=new Node
      newNode->data=1
      newNode->next=NULL
    //connecting playlist to main list node
     newNode_music->down=newNode

  //case where a new song must be inserted in an existing playlist [4]
   else
       tmp2=tmp //tmp2 used to traverse the playlist
       while(tmp2->next!=tmp) //finding the last element in the playlist
           tmp2=tmp2->next
```

```
        Node newNode=new Node
        newNode->data=tmp2->data+1
        newNode->next=tmp
        tmp2->next=newNode
end function
```

(c) Write the pseudocode of the function DELETE_SONG(head,genre,x) that deletes song number x from the playlist corresponding to the music genre ‘genre’. If music genre ‘genre’ or song x in the corresponding music genre do not exist, a message should be displayed on screen. [10]

Answer:

```
function DELETE_SONG(head,genre,x)
   //checking empty main list [1]
   if(head==NULL)
         print(“Empty main list”)
         return
    //getting the pointer to the correct playlist [1]
   tmp=SEARCH_MUSIC_TYPE(head,genre)

  //music genre ‘genre’ does not exist in main list [1]
  if(tmp==NULL)
     print(“No playlist for this music genre”)
  else
    //tmp will be used to mark the start of the playlist
     tmp1=tmp //pointer to traverse the playlist [1]
     prev1=tmp1 //prev1 will be used in the delete operation [1]
     while(tmp1->data!=x) //loop checking song does exist [1]
          prev=tmp1 //move to the next song [0.5]
          tmp1=tmp1->next  //move to the next song [0.5]
          if(tmp1==tmp) //whole playlist traversed, song not found [2]
             print(“Song not found in playlist”)
             return
    //song found, tmp1 points to the song and prev to the previous song
    prev->next=tmp1->next //we delete song pointed by tmp1 [1]
    free(tmp1) //optional, it depends on the programming language
end function
```

Question 4

A software developer needs to solve the following problem: given the adjacency matrix of small social network of friends, find the k-th most popular person (that is, the k-th person with the highest number of friends). Assume that the graph is undirected (that is, friendship is reciprocal). A number 1 in the adjacency matrix signals the existence of friendship and a number 0 signals the absence of friendship.

For example, for the graph represented by the following adjacency matrix M:

	[0]	**[1]**	**[2]**	**[3]**
[0]	0	1	1	1
[1]	1	0	1	0
[2]	1	1	0	0
[3]	1	0	0	0

The 1st most popular person (that is, k=1) is 0, with 3 friends. The second most popular person (k=2) is 1 and 2, with 2 friends each and the third most popular person (k=3) is 3, with only one friend.

The algorithm to design must take as input arguments the adjacency matrix (M), its number of nodes (N) and the value of k. It must return the identity (number) of the k-th most popular person in the network.

The software developer came up with these two algorithms to solve the problem:

Algorithm 1:

1. Create a N-element array, A, storing the number of friends of each person in the social network. Element 0 in array A stores the number of friends of person 0, element 1 in array A stores the number of friends of person 1 and so on.
2. Create a variable, called max, where the identity of the person with the highest number of friends will be recorded
3. Create a variable, called count, initialised to 0
4. Visit every element of A and record the index of the maximum value in max
5. Once the maximum value is found, increase count by one unit
6. If count==k, return the value of max. Otherwise, set the value of position max to zero in array A and repeat steps 2-6

Algorithm 2:

- Create a max-heap, where each node stores two numbers: the number of friends of a person and the identity of the person. The number of friends determines how the max-heap is built. Thus, the node (person) with the highest number of friends will be stored in the root of the max-heap.

- Perform EXTRACT_MAX k times. The node last extracted stores the identity the k-th most popular person.

(a) Write the pseudocode of Algorithm 1 [8]

Answer:

```
function A1(M,N,k)
  //creating array A
   for 0<= i < N
     sum=0
     for 0<= j <N
         sum=sum+M[i,j]
     A[i]=sum
//initialise count
count=0
  //while loop to keep record of number of max found
   while(count<k)
     //find mmax in A
      max=0
      for 1<= i < N
          if(A[i]>A[max])
               max=i
      end for
     //delete found max
      A[max]=0
    //update count  [1]
     count=count+1
  end while
  //return k-th min value [1]
  return max
```

(b) Write the pseudocode of Algorithm 2. [8]

Assume you already have implemented the following modified max-heap functions:

- MOD_INSERT(heap,x, y): insert the pair (x,y) into the heap: x corresponds to the number of friends and y to the identity of the person. Worst-case Theta(logN)
- MOD_EXTRACT_MAX(heap): returns the identity of the person with the maximum number of friends stored in the max-heap, eliminates that node and re-arranges the heap to meet the shape and heap properties. Worst-case Theta(logN)

Answer:

```
function A2(M,N,k)
  //creating array A [1]
   for 0<= i < N
     sum=0
     for 0<= j <N
         sum=sum+M[i,j]
     end for
     MOD_INSERT(heap, sum, i)
   end for
  //extract the min values [2]
   for 1<= i <=k
       max=EXTRACT_MAX(heap)
 //return the k-th min value [2]
  return max
end function
```

(c) What are the worst-case time complexities of A1 and A2? Use Theta-notation. Justify your findings. [8]

Answer:

A1 running time can be expressed as $C_0*N^2+C_1*k*N+C_2$. Thus, irrespective of whether k is in the order of N or much less than N, A1 is Theta(N^2).
A2 running time can be expressed as $C_0*N^2+ C_1*N\log N+ k*C_2*\log N$. Thus, irrespective of whether k is in the order of N or much less than N, A2 is Theta(N^2).

(c) What algorithm do you recommend for implementation? Justify [6]

Answer

- Any of the can be implemented, as they have the same worst-case time

END OF PAPER