SLIATE

SRI LANKA INSTITUTE OF ADVANCED TECHNOLOGICAL EDUCATION

(Established in the Ministry of Higher Education, vide in Act No. 29 of 1995)

Higher National Diploma in Information Technology
First Year, Second Semester Examination-2017
HNDIT1211: Data Structures and Algorithms

Instructions for Candidates:
Answer four (04) questions only.

No. of questions : 0
No. of pages : 0
Time : Two hours

Model Answers

Question 01

a. What is Data Structure? (03 Marks)

Data structure is the arrangement of data in a computer memory/storage.

b. State whether the following data structures leaner or nonlinear. (04 Marks)

i. Tree - Nonlinear
ii. Stack - Linear
iii. Queue - Linear
iv. Graph - Nonlinear

c. Briefly explain the following terms. (05 Marks)

i. Abstract Data Types (ADT)

ADT is a specification of a mathematical set of data and the set of operations that can be performed on the data.

ii. Big O Notation

We can say that a function is "of the order of n", which can be written as O(n) to describe the upper bound on the number of operations. This is called Big-Oh notation.

iii. Best Case Efficiency

Best case efficiency is the minimum number of steps that an algorithm can take any collection of data values.

iv. Worst Case Efficiency

Worst case efficiency is the maximum number of steps that an algorithm can take for any collection of data values.

v. Average Case Efficiency

Average case efficiency

- is the efficiency averaged on all possible inputs
- must assume a distribution of the input
- is normally assumed for uniform distribution (all keys are equally probable)

d. Write C++ code to initialize an array with a word having seven (7) characters and print whether the word is palindrome or not. [NOTE: A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward as forward, such as 'madam' or 'tacocat'] (06 Marks)

```
#include<iostream.h>
void main()
{
    int f, r;
    int palindrome=1; //for boolean
    char word[7]={'R','A','C','E','C','A', 'R'};
    for (f=0, r=6; f<4 && palindrome; f++, r--)
        if (word[f] != word[r]) palindrome=0;
    if palindrome
        cout<<"Palindrome"
    else
        cout<<"Not Palindrome";
}
```

Or

```
#include<iostream.h>
void main()
```

```
{
 int f, r;
 char word[7]={'R','A','C','E','C','A', 'R'};
 for (f=0, r=6; (f<4) && (word[f] == word[r]); f++, r--)
 if (f==4)
     cout<<"Palindrome"
 else
     cout<<"Not Palindrome";
}
```

e. Write C++ code to create the following two dimensional array and display its elements.

5	3	5
8	4	2
2	6	3
7	4	1

(07 Marks)

```
#include<iostream.h>
void main()
{
Int i,j;
int matrix[4][3]={{5,3,5},{8,4,2},{2,6,3},{7,4,1}};
cout<<"The elements of the array are";
for (i=0; i<4; i++)
{
for (j=0; j<3; j++)
cout<<matrix[i][j]<<"\t";
cout<<"\n";
}
}
```

Question 02

a. Define Stack Data Structure and mention two (02) ways of implementing Stack. (03 Marks)

Stack is a data structure which is used to handle data in a last-in-first-out (LIFO) method.

Static (Array based) Implementation

Dynamic (Linked List based) Implementation

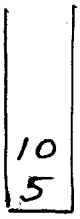
b. Graphically illustrate the following stack operations sequentially. (04 Marks)

i. initializeStack()

ii. push(5)

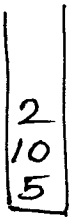
iii. push(10)

iv. a = topEement( )

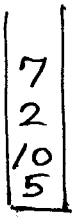


a=10

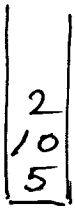
v. push(2)



vi. push(7)



vii. b = pop()



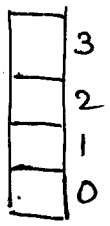
b=7

viii. displayElements()

2
10
5

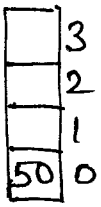
c. Graphically illustrate the implementation of the following stack operations. (05 Marks)

i. initializeStack()



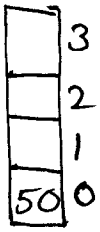
top = -1

ii. push(50)



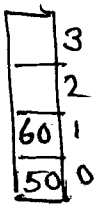
top = 0

iii. y = topElement( )



top = 0
y = 50

iv. push(60)



top = 1

v. x = pop()

top = 0
x = 60

d. Carefully read and understand the following class definition for implementing a stack data structure.

```
const STK_SIZE=5;
class Stack
{
private:
   int top;
   int stk[STK_SIZE];
public:
      ...
      void push(int);
      int pop();
      int topElement();
      ...
}
```

Now write C++ code to implement the following operations. (06 Marks)

i. Adding a new element in on top of the stack.

```
void Stack::push(int elt)
{
   if (top < STK_SIZE-1) stk[++top]=elt;
}
```

ii. Removing the element from the top of the stack.

```
int Stack::pop()
{
   if (top > -1)
     return stk[top--];
   else
     return 999; //Some invalid integer should be returned
}
```

iii. Referring the top element of the stack without removing it.

```
int Stack::topElement()
{
   if (top > -1)
     return stk[top];
   else
     return 999; //Some invalid integer should be returned
};
```

e. Carefully observe the following output. Here user input is formatted as bold and underlined.

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: **1**
Enter an element to be pushed: **50**

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: **1**
Enter an element to be pushed: **25**

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: **2**

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: **1**
Enter an element to be pushed: **60**

1.Push 2.Pop 3.Display 4.Exit
Enter your choice: **3**
60 50

1.Push 2.Pop 3.Display 4.Exit

Enter your choice: **4**

Considering that the Stack class has been completely defined, write only the main method to run to output as shown in the above sample output. (07 Marks)

```
void main()
{
   int ch;
   int x;
   Stack st;
   while(1)
     {
        cout <<"\n1.Push  2.Pop  3.Display  4.Exit\nEnter your choice: ";
        cin >> ch;
        switch(ch)
         {
          case 1:  cout <<"Enter the element to be pushed: ";
                   cin >> ch;
                   st.push(ch);
                   break;
          case 2:  x=st.pop();
                   cout<<x<<" has been popped out."<<endl;
                   break;
          case 3:  st.display();break;
          case 4:  exit(0);
         }
     }
}
```

Question 03

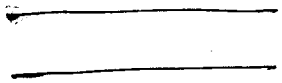
a. How a queue differ from stack in operation and in static implementation? (03 Marks)

- In operation, stacks function in Last-In-First-Out method while queues function in First-In-First-Out method. (01 Marks)
- In implementation, for stack it is enough to one variable top, but, for queue three variables, namely front, rear and size, are necessary.

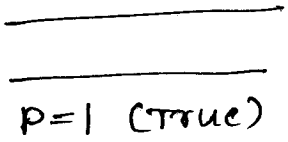
  Further in stack, top can be incremented/decremented linearly, but, in queue the front and rear should be incremented rotationally. (02 Marks)

b. Diagrammatically illustrate the following Queue Operations. (04 Marks)

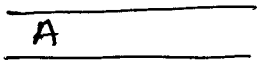
i. initializeQueue()



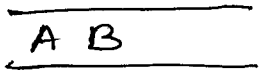
ii. p = isEmpty()



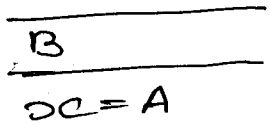
iii. enQueue(A)



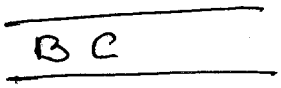
iv. enQueue(B)



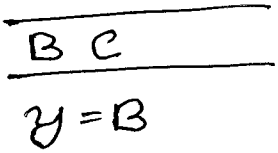
v. x = deQueuer()



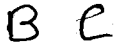
vi. enQueue(C)



vii. y = frontElement()

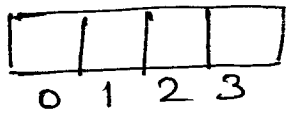


viii. displayQueue()



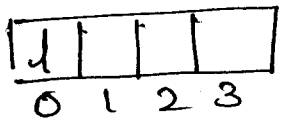
c. Diagrammatically illustrate the implementation of the following queue operations. (05 Marks)

i. initializeQueue()



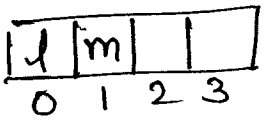
front = 0
rear = 0
size = 0

ii. enQueue(l)



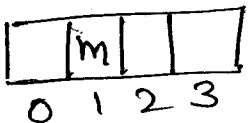
front = 0
rear = 1
size = 1

iii. enQueue(m)



front = 0
rear = 2
size = 2

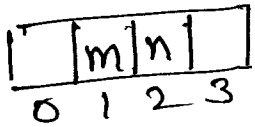
iv. x = deQueue()



x = l

front = 1
rear = 2
size = 1

v. enQueue(n)



front = 1
rear = 3
size = 2

d. Carefully read and understand the following class definition for the Queue data structure.

```
#include<iostream.h>
const Q_SIZE=5;
class Queue
{
private:
 int front, rear, size;
 int que[Q_SIZE];
public:
 ...
```

```
void enQueue(int);
int deQueue();
void displayQueue();
...
}
```

Now write code to implement the methods for the following operations. (06 Marks)

i. Inserting an item to the queue

```
void Queue::enQueue(int elt)
{
if (size < Q_SIZE)
{
rear=(rear+1)%Q_SIZE;
que[rear]=elt;
size++;
} //Else cout<<"Queue is full"
}
```

ii. Removing an item from the queue

```
int Queue::deQueue()
{
if (size > 0)
{
front=(front+1)%Q_SIZE;
size--;
return que[front];
}
else
return 999; //Some invalid integer should be returned or cout<<"Queue is empty"
}
```

iii. Displaying all the items in the queue

```
void Queue::displayQueue()
{
 int i=front;
 for (int j=1;j<=size;j++)
 {
 i=(i+1)%Q_SIZE;
 cout<<que[i]<<endl;
 }
}
```

e. Match the following Linked List Operations with the corresponding Queue operations.

	Linked List Operations		***Queue Operation***
i.	deleteFirstElement()	A	initializeQueue()
ii.	firstElement()	B	enQueue()
iii.	initializeList()	C	deQueue()
iv.	isEmpty()	D	frontElement()
v.	isFull()	E	isEmpty()
vi.	displayList()	F	isFull()
vii.	insertAtRear()	G	displayQueue()

(07 Marks)

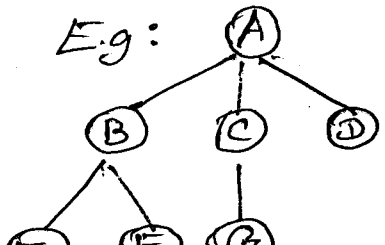
i. C
ii. D
iii. A
iv. E
v. F
vi. G
vii. B

Question 04

a. Briefly explain the following terms related to tree data structure with the help of diagrams. (03 Marks)

i. Tree

A tree is a data structure that emulates a hierarchical tree structure with a set of linked nodes.



ii. Root of a Tree

There is one unique node in every tree which has no parent and is called the root of the tree.

E.g: (A) is the root of the above tree.

iii. Parent of a Node

Every child node has a unique parent.

E.g: (A) is the parent of (B), (C), and (D)
B is the parent of (E) and (F)

iv. Children of a Node

Every parent node can have any number of children (including none).

E.g: (B), (C) and (D) are children of (A)
(E) and (F) are children of (B)

v. Siblings of a Node

Nodes with the same parent are called siblings.

E.g: (C) and (D) are siblings of (B)
(E) is sibling of (F)

vi. Leaves of a Tree

A leaf node has no children.

E.g: (E), (F), (G) and (D) are leaves of the above tree

b. Define the following terminologies related to tree data structure. (04 Marks)

i. Size of a Tree

The size of a tree is the number of nodes that it contains.

ii. Depth of a Node

The depth of a node is the number of edges from the root to the node.

iii. Degree of a Tree

The degree of a node is the number of its children.

The degree of a tree is the maximum degree of any of its nodes.

iv. Path between two nodes.

Path between two nodes in a tree is a sequence of edges which connect those nodes.

c. Explain the following special trees

i. Binary Tree (02 Marks)

A binary tree is a rooted tree in which no node can have more than two children.

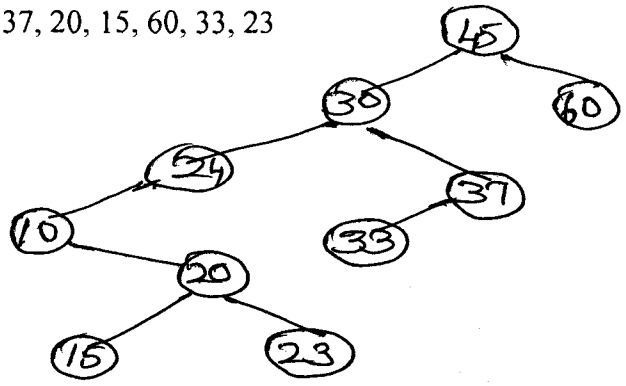
ii. Binary Search Tree (03 Marks)

It is a binary tree such that for every node N in the tree:

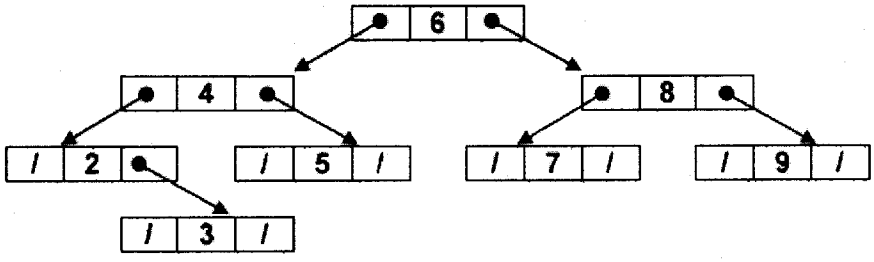
- All keys in N's left sub tree are less than the key in N, and
- All keys in N's right sub tree are greater than the key in N.

d. Insert the following data set into a binary search tree.

45, 30, 24, 10, 37, 20, 15, 60, 33, 23 (06 Marks)



e. Diagrammatically explain the implementation of a Binary Tree. (07 Marks)



Question 05

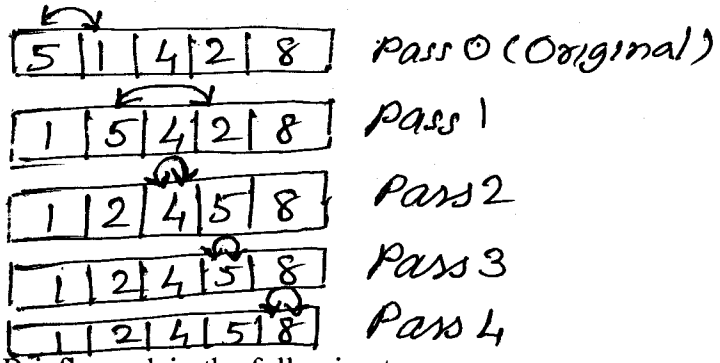
a. Give a single line explanation for the following terms. (03 Marks)

i) Sorting: Arranging items in ascending or descending order is called as sorting.

ii) Selection Sort: Here we repeatedly find the next largest (or smallest) element in the array and move it to its final position in the sorted array.

iii) Bubble Sort: Here we repeatedly move the largest element to the highest index position of the array.

b. Sort the data set 5, 1, 4, 2, 8 using Selection Sort method. Show your work (04 Marks)



c. Briefly explain the following terms.

i. Searching (01 Marks)

A search algorithm is an algorithm for finding an item among a collection of items.

ii. Sequential Searching (02 Marks)

It examines the first element in the list and then second element and so on until a much is found.

iii. Binary Searching (02 Marks)

Here the elements should be in (ascending) order and the elements should be saved in a randomly accessible data structure like array.

The basic algorithm is to find the middle element of the list, compare it against the key/target, decide which half of the list must contain the key, and repeat with that half.

d. Give C++ implementation for Sequential Search. (06 Marks)

```
int sequentialSearch(int *a, int n, int t)
{
    int i;
    for (i = 0; i < n; i++)
        if (a[i]==t) return i;
    return (-1);
}
```

e. Write recursive pseudo code for Binary Search. (07 Marks)

```
int recBinarySearch(a[],l,u,t) //It returns the location of t in the array a[] from the index l to u.
 if l>u then
    return -1
 else
    mid=(l+u)/2
    if t=a[mid] then
        return mid
    else if t<a[mid] then
        return recBinarySearch(a[],l,mid-1,t)
    else
        return recBinarySearch(a[],mid+1,u,t)
    end if
 end if
```


Thathcyan 0773659589