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-2016
HNDIT1211: Data Structures and Algorithms

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

No. of questions : 05
No. of pages : 04
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. Compare Linear and Nonlinear Data Structures by giving two (02) examples for each Data Structure. (04 Marks)

Linear Data Structure	***Nonlinear Data Structure***
Data Organized Sequentially (one after the other)	Data organized non sequentially
Easy to implement because the computer memory is also organized as linear fashion	Difficult to implement
E.g: Array, Stack, Queue, Linked List	E.g: Tree, Graph

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

i. Algorithm

An algorithm is a step by step procedure for solving a problem in a finite amount of time.

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 10 characters (A…J) and print them in reverse order. (06 Marks)

```
#include<iostream.h>
void main()
{
int i;
char keys[10]={'A','B','C','D','E','F', 'G','H','I',’J’};
for (i=9; i>=0; i--) cout<<keys[i]<<" ";
}
```

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

5	3
8	4
2	6
7	4

(07 Marks)

```
#include<iostream.h>
void main()
{
Int i,j;
int matrix[4][2]={{5,3},{8,4},{2,6},{7,4}};
```

```
cout<<"The elements of the array are";
for (i=0; i<4; i++)
{
for (j=0; j<2; 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(a)

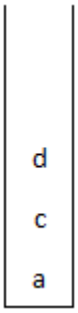
iii. push(b)

iv. x = pop( )

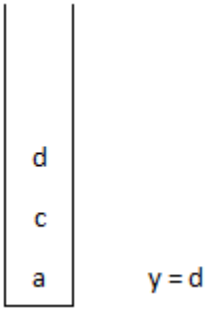
x = b

v. push(c)

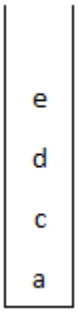
vi. push(d)



vii. y = topElement()

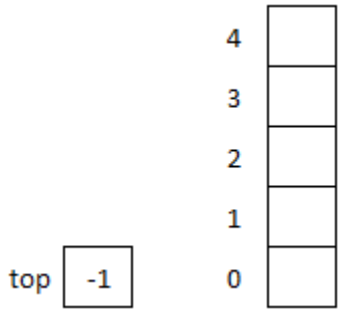


viii. push(e)

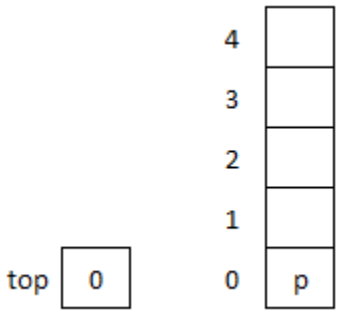


c. Graphically illustrate the implementation of the following stack operations. (05 Marks)

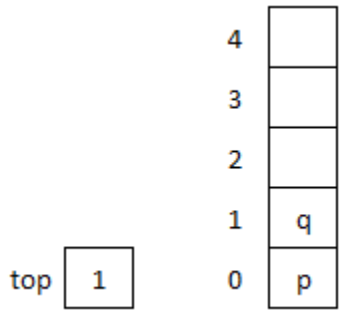
i. initializeStack()



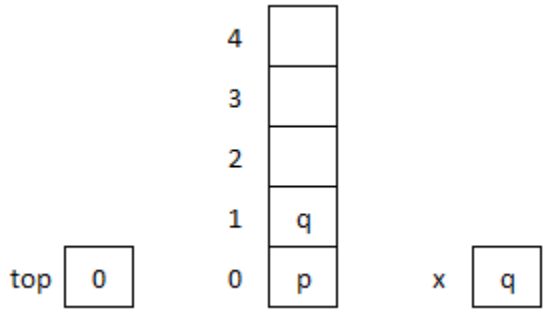
ii. push(p)



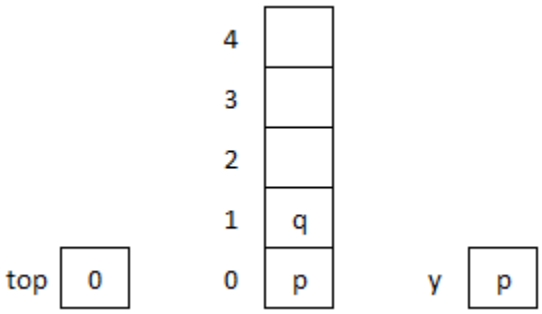
iii. push(q)



iv. x = pop()



v. y = topElement( )



d. Write down C++ code for implementing the following operations in stack. (06 Marks)

```
const STK_SIZE=5;
class Stack
{
private:
 int top;
 int stk[STK_SIZE];
public:
```

```
 …
 void push(int);
 int pop();
 int isEmpty();
 …
}
```

i. push( )

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

ii. pop( )

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

iii. isEmpty()

```
int Stack::isEmpty()
{
 return (top == (-1));
}
```

e. Match the following stack operations with the corresponding linked list operations.

	Stack Operation		***Linked List Operations***
i.	initializeStack()	A	insertAtFront()
ii.	push()	B	firstElement()
iii.	pop()	C	initializeList()

iv.	topElement()	D	isEmpty()
v.	isEmpty()	E	deleteFirstElement()
vi.	isFull()	F	displayList()
vii.	displayStack()	G	isFull()

(07 Marks)

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

Question 03

a. Queue is an abstract data type (ADT). Explain briefly (03 Marks)

Queue specifies data stored and the operations on the data.

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

i. initializeQueue()

ii. p = isEmpty()

p = true

iii. enQueue(A)

A

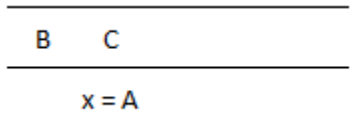
iv. enQueue(B)

A B

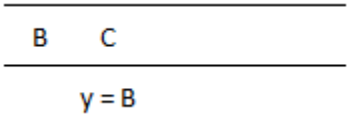
v. enQueue(C)

A B C

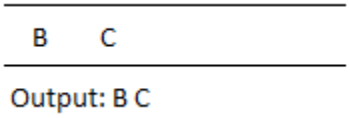
vi. x = deQueuer()



vii. y = frontElement()

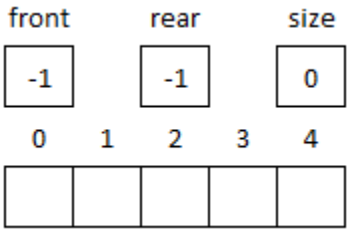


viii. displayQueue()

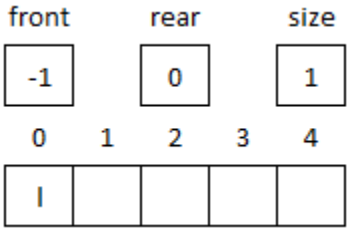


c. Diagrammatically illustrate the implementation of the following queue operations. (05 Marks)

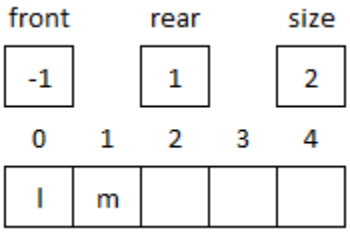
i. initializeQueue()



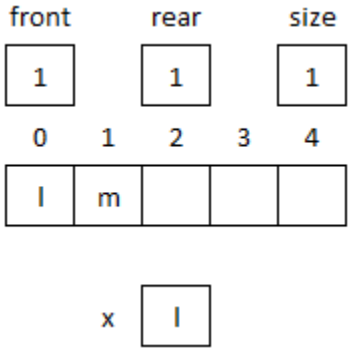
ii. enQueue(l)



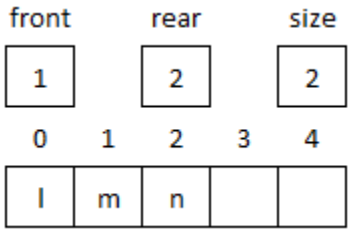
iii. enQueue(m)



iv. x = deQueue()



v. enQueue(n)



d. Write C++ code for the following functions of queue (06 Marks)

```
#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();
…
}
```

i. Inserting an item to a 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 a 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 items of a 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 Queue operations with the corresponding Linked List Operations.

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

(07 Marks)

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

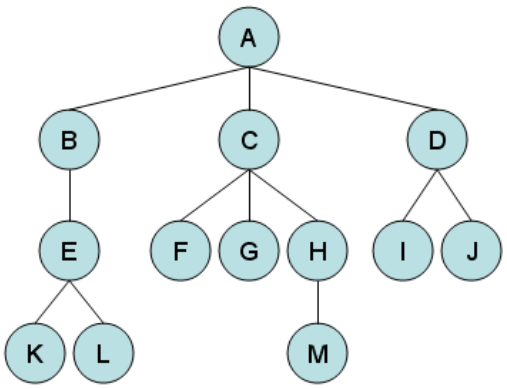
Question 04

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.

E.g.:



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.: The root of the above tree is A.

iii. Parent of a Node

Every child node has a unique parent.

E.g.: In the above tree, B is the parent of E.

iv. Children of a Node

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

E.g.: In the above tree, I and J are children of D.

v. Siblings of a Node

Nodes with the same parent are called siblings.

E.g.: In the above tree, I and J are siblings.

vi. Leaves of a Tree

A leaf node has no children.

E.g.: In the above tree, K L F G M I and J are leaves.

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. Height of a Tree

The height of a tree is the largest depth of any of its nodes.

iv. Degree of a Node

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

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.

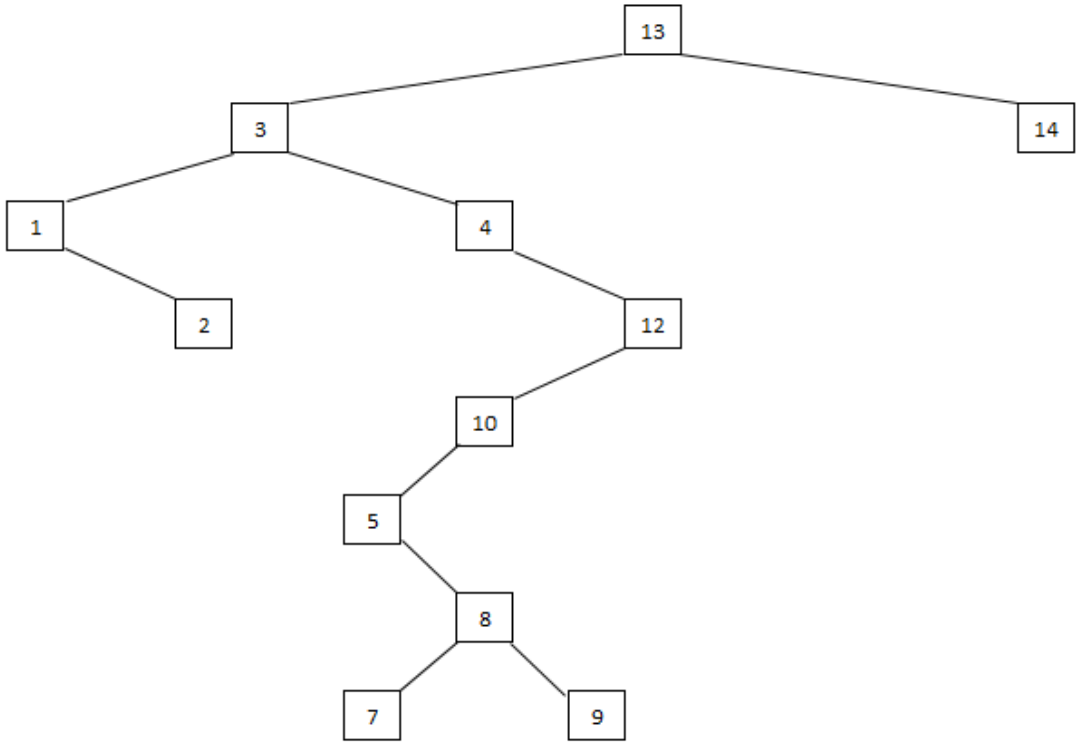
ii. Binary Search Tree (03 Marks)

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

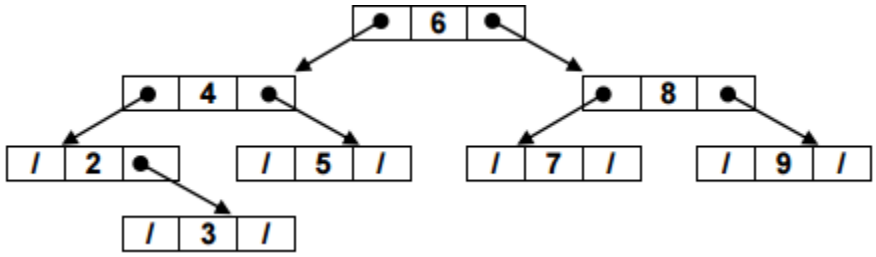
- All keys in N's left sub tree are less than the key in N, and
- All keys in N's right sub tree are greater than the key in N.

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

13, 3, 4, 12, 14, 10, 5, 1, 8, 2, 7, 9 (06 Marks)



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



Question 05

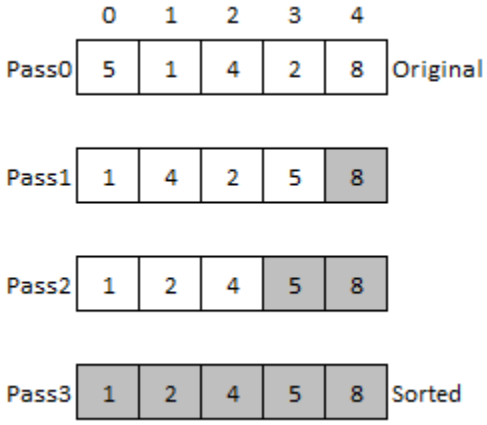
a. What is meant by sorting?

Give two (2) sorting algorithms other than Bubble Sorting. (03 Marks)

Arranging items in ascending or descending order is called as sorting.

Selection Sort, Quick Sort, etc.

b. Sort the following data set 5, 1, 4, 2, 8 using Bubble 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. Give C++ implementation for Binary Search. (07 Marks)

```
int binarySearch(int* a, int l, int u, int t)
{
 int p;
 p = ( l + u) / 2;
 while((a[p] != t) && (l<=u))
 {
 if (a[p] > t)
 u = p - 1;
 else
 l = p + 1;
 p = (l + u) / 2;
 }
 if (l <= u)
 return p;
 else
 return (-1);}
```