

Department of Information & Communication Technology
MIT, Manipal

III Sem B.Tech (IT/CCE)

Data structures (ICT 2153), In-sem Examination

Date: 14/12/2021

Max. Marks: 20

1	<p>A doubly linked list is used to store item number, quantity, price per item and seller details. Write the class definition with suitable data members and member functions to perform the following operations on doubly linked list.</p> <ul style="list-style-type: none">a. Update the quantity after reading item number and number of items from the user. [1m]b. Delete alternate nodes from the doubly linked list [1.5m] <p>//class definition(0.5m)</p> <pre>class cart{ int item_no; int quantity; float price; char seller_details[20]; public: void update(); void deleteNode(cart *) void deleteAltNodes(cart *); }; void cart::update() { int n;cart *c=first; cout<<"Enter item number to be updated: "; cin>>n; while(c) { if(c->item_no==n){ cout<<"Enter new item quantity value: "; cin>>c->quantity; cout<<"Quantity updated successfully\n"; } c=c->next; } } void cart::deleteNode(cart* del) { if (first==NULL del == NULL) return; if (first == del) first=del->next; if (del->next != NULL) del->next->prev = del->prev; if (del->prev != NULL) del->prev->next = del->next; delete(del); return; } cart* cart::deleteAltNodes(cart *f)</pre>	3
----------	--	----------

	<pre>{ cart *ptr = f; cart *n; int c=0; while (ptr != NULL) { n = ptr->next; c=c+1; if(c%2==0) deleteNode(ptr); ptr = n; } return(f); }</pre>			
2	<p>Define a class to represent a sparse matrix in the array of object form. Write a user defined function to delete negative element from the sparse matrix represented as array of objects. Assume that the sparse matrix represented as array of objects is passed as an argument to the delete function.</p> <p>(Class definition: 0.5M; reading key ele: 0.5M; traversing, checking and deleting: 1.5M; Not found case: 0.5M))</p> <table><tr><td><pre>void spm::spm_del_neg(spm a[]) { int ele, flag = 0; cout<<"Enter the element to delete: \n"; cin>>ele; if(ele<0){ for(int i=1;i<a[0].val;i++) { if(ele != a[i].val) continue; for(int j = i; j<a[0].val;j++) { flag = 1; a[j] = a[j+1]; } a[0].val = a[0].val-1; } } }</pre></td><td><pre>else if(ele>=0) { cout<<"Element is not a negative number! \n"; } if(flag = 0) { cout<<"Element not found! \n"; } }</pre></td></tr></table>	<pre>void spm::spm_del_neg(spm a[]) { int ele, flag = 0; cout<<"Enter the element to delete: \n"; cin>>ele; if(ele<0){ for(int i=1;i<a[0].val;i++) { if(ele != a[i].val) continue; for(int j = i; j<a[0].val;j++) { flag = 1; a[j] = a[j+1]; } a[0].val = a[0].val-1; } } }</pre>	<pre>else if(ele>=0) { cout<<"Element is not a negative number! \n"; } if(flag = 0) { cout<<"Element not found! \n"; } }</pre>	3
<pre>void spm::spm_del_neg(spm a[]) { int ele, flag = 0; cout<<"Enter the element to delete: \n"; cin>>ele; if(ele<0){ for(int i=1;i<a[0].val;i++) { if(ele != a[i].val) continue; for(int j = i; j<a[0].val;j++) { flag = 1; a[j] = a[j+1]; } a[0].val = a[0].val-1; } } }</pre>	<pre>else if(ele>=0) { cout<<"Element is not a negative number! \n"; } if(flag = 0) { cout<<"Element not found! \n"; } }</pre>			
3	<p>Write a complete class definition with data member and member functions to perform insert and delete operations on i^{th} queue in an array of multiple linear queues. Write the circular queue content (max size: 4), values of front and rear for each of the following operations on circular queue. Also, specify the initial values of front and rear.</p> <p>i. Insert (9) ii. Insert (23) iii. Insert (67) iv. Delete v. Delete vi. Insert (56) vii. Insert (45) viii. Insert (12)</p> <p>[multiple queue class: 0.5, enqueue: 0.5m, deque: 0.5m, constructor: 0.5m, Circular Queue content: 1m]</p> <pre>class mqueue { int a[50], bottom[10], f[10],r[10], maxsize, nq; public: mqueue();</pre>	3		

```

mqueue(int,int);
void enQ(int,int);
int deQ(int);
void disp(int);
};
mqueue::mqueue()
{
maxsize=10;
nq=1;
}
mqueue::mqueue(int m,int n)
{
maxsize=m;
nq=n;
for(int i=0;i<nq;i++)
f[i]=r[i]=bottom[i]=(maxsize/nq)*i-1;
}
void mqueue::enQ(int ele,int i)
{
if(r[i]==bottom[i+1]||r[i]==maxsize-1)
cout<<"\nqueue is full";
else
a[++r[i]]=ele;
}
int mqueue::deQ(int i)
{
if(f[i]==r[i]){
cout<<"\nqueue is empty";
return -1;
}
else return(a[++f[i]]);
}

```

front=0; rear=0

i. Insert(9): front=0;rear=1

	9		
--	---	--	--

ii. Insert(23) front=0;rear=2

9	23		
---	----	--	--

iii. Insert(67) front=0;rear=3

9	23	67	
---	----	----	--

iv. Delete front=1;rear=3

	23	67	
--	----	----	--

Deleted element is 9

v. Delete front=2;rear=3

		67	
--	--	----	--

Deleted element is 23

vi. Insert(56) front=2;rear=0

56			67
----	--	--	----

vii. Insert(45) front=2; rear=1

56	45		67
----	----	--	----

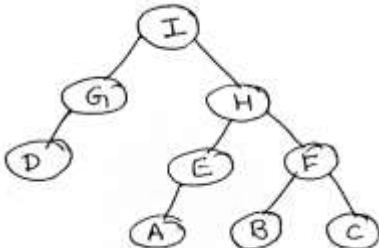
viii. Insert(12) Q is FULL			
56	45		67
Circular Q is Full. Can't insert 12 into queue			
4	Write a user defined function to find a string with maximum number of vowels in it. Input to the function is the set of strings. Find vowelcount in each string: 2 marks, Find string with maximum vowels: 1 mark		
<pre>void max(char str[10][10], int n, char r[10]){ int vowcount[n]; char vowels[]={"AEIOUaeiou"}; int i, j, maxcount, k; for(i=0;i<n;i++) vowcount[i]=0; for(i=0;i<n;i++) for(j=0;j<strlen(str[i]);j++) for(k=0;k<strlen(vowels);k++) if(str[i][j]==vowels[k]) vowcount[i]++;</pre>		<pre>maxcount=vowcount[0]; k=0; for(i=1;i<n;i++){ if(vowcount[i]>maxcount){ maxcount=vowcount[i]; k=i;} } strcpy(r, str[k]); }</pre>	
5	Write the inorder, postorder, preorder and level order traversal sequence for the tree shown in Figure 1. Write recursive user-defined functions to a. Count and return the total number of nodes with degree one (1m)		
<pre>void tree::disp(node *r ,int *co1) { if(r) { disp(r->lchild,co0,co1,co2); cout<<r->data; if((r->lchild==NULL && r->rchild!=NULL) && (r->lchild!=NULL r->rchild==NULL) (*co1)++; disp(r->rchild,co0,co1,co2); } }</pre>			
b. Display the nodes of only the right subtree of the binary tree (1m)			
			

Figure 1

```
void tree::inorderr(node *ptr)
{
    static node *key=ptr;
    if(ptr)
    { inorderr(ptr->rchild);
      cout<<ptr->data<<" ";
      if (ptr==key) return;
      inorderr(ptr->lchild);
    }
}
```

[inorder: 0.5m, preorder: 0.5m, postorder:0.5m, level order:0.5m]

Inorder: DGIAEHBFC

Preorder: IGDHEAFBC

Postorder: DGAEBCFHI

Level order: IGHDEFABC

6 (a) Write a user function named **arrange**. The arrange () should sort the nodes according to the length of the string that they store. For example, if nodes contain “This” -->”is” -->”an” -->”example” then it should be rearranged as “is” -->”an” -->” This” -->”example”

4

(b) Write a user defined function to split a singly linked list into two halves. If the number of elements are odd, the first half should contain one element more.

((a) Traverse :0.5; Comparing: 0.5M; Sorting:1M

(b)) Determine the midpoint: 0.5 mark, split: 1.5 marks.

(a)
node* node::sort1(node *head)
{
 node *t, *cur = head;
 int n;
 char arr[25];
 while(cur->next!=NULL)
 {
 t = cur->next;
 while(t!=NULL)
 {
 if(strlen(cur->info) > strlen(t->info))
 {
 strcpy(arr, cur->info);
 strcpy(cur->info, t->info);
 strcpy(t->info, arr);
 }
 }
 }
}

(b)
node* node::split(node * head)
{
 node *list1, *list2,*cur;
 int count=0, halfcount=0, i;
 cur=head;
 while(cur!=NULL)
 {
 count++;
 cur=cur->next;
 }
 halfcount=count/2;
 cur=head;
 for(i=1;i<=halfcount;i++)
 {
 list1=cur;
 cur=cur->next;
 }
}

	<pre>t = t->next; } cur = cur->next; } return head; }</pre>	<pre> } if(count%2==1){ list1=cur; list2=cur->next; list1->next=NULL; } else{ list2=cur; list1->next=NULL;} list1=head; return list2; }</pre>	
--	---	--	--