

Department of Information & Communication Technology

MIT, Manipal

III Sem B.Tech (IT/CCE)

Data structures (ICT 2153), In-sem Examination scheme (set2)

Date: 14/12/2021

Max. Marks: 20

1	<p>Define a class to represent a sparse matrix in the array of object form. Write a user defined function to delete an element from the sparse matrix represented as array of object. 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(spm a[]) { int ele, flag = 0; cout<<"Enter the element to delete: \n"; cin>>ele; for(int i=1;i<a[0].val;i++) { if(ele != a[i].val) continue; flag = 1; for(int j = i; j<a[0].val;j++) { a[j] = a[j+1]; } } a[0].val = a[0].val-1;</pre></td><td><pre>if(flag = 0) { cout<<"Element not found! \n"; } }</pre></td></tr></table>	<pre>void spm::spm_del(spm a[]) { int ele, flag = 0; cout<<"Enter the element to delete: \n"; cin>>ele; for(int i=1;i<a[0].val;i++) { if(ele != a[i].val) continue; flag = 1; for(int j = i; j<a[0].val;j++) { a[j] = a[j+1]; } } a[0].val = a[0].val-1;</pre>	<pre>if(flag = 0) { cout<<"Element not found! \n"; } }</pre>	3
<pre>void spm::spm_del(spm a[]) { int ele, flag = 0; cout<<"Enter the element to delete: \n"; cin>>ele; for(int i=1;i<a[0].val;i++) { if(ele != a[i].val) continue; flag = 1; for(int j = i; j<a[0].val;j++) { a[j] = a[j+1]; } } a[0].val = a[0].val-1;</pre>	<pre>if(flag = 0) { cout<<"Element not found! \n"; } }</pre>			
2	<p>Write a user defined function that takes prefix expression as an argument, evaluates, and displays the result. Show all the steps, with stack contents, involved in converting + A / * B % - C D E F to fully parenthesized infix expression.</p> <p>[prefix evaluation code: 2m, Converting prefix to fully parenthesized infix: 1m]</p> <table><tr><td><pre>class STACK {int TOP; int a[30]; public: STACK(){TOP=-1;} void push(int); int pop(); }; void STACK::push(int ele) { if (TOP == 29) cout<<"stack is full; else a[++TOP]=ele; } int STACK::pop() { if(TOP== -1) return -999; else return a[TOP--]; } void prefix_evaluation() {</pre></td><td><pre>if(symbol >= 48 && symbol<= 57) S.push(symbol-'0'); else { for(j=0;j<2;j++) { a[j]=S.pop(); } switch(symbol) { case '%': c=a[0]%a[1]; S.push(c); break; case '/': c=a[0]/a[1]; S.push(c); break; case '*': c=a[0]*a[1]; S.push(c); break; case '+': c=a[0]+a[1]; S.push(c);</pre></td></tr></table>	<pre>class STACK {int TOP; int a[30]; public: STACK(){TOP=-1;} void push(int); int pop(); }; void STACK::push(int ele) { if (TOP == 29) cout<<"stack is full; else a[++TOP]=ele; } int STACK::pop() { if(TOP== -1) return -999; else return a[TOP--]; } void prefix_evaluation() {</pre>	<pre>if(symbol >= 48 && symbol<= 57) S.push(symbol-'0'); else { for(j=0;j<2;j++) { a[j]=S.pop(); } switch(symbol) { case '%': c=a[0]%a[1]; S.push(c); break; case '/': c=a[0]/a[1]; S.push(c); break; case '*': c=a[0]*a[1]; S.push(c); break; case '+': c=a[0]+a[1]; S.push(c);</pre>	3
<pre>class STACK {int TOP; int a[30]; public: STACK(){TOP=-1;} void push(int); int pop(); }; void STACK::push(int ele) { if (TOP == 29) cout<<"stack is full; else a[++TOP]=ele; } int STACK::pop() { if(TOP== -1) return -999; else return a[TOP--]; } void prefix_evaluation() {</pre>	<pre>if(symbol >= 48 && symbol<= 57) S.push(symbol-'0'); else { for(j=0;j<2;j++) { a[j]=S.pop(); } switch(symbol) { case '%': c=a[0]%a[1]; S.push(c); break; case '/': c=a[0]/a[1]; S.push(c); break; case '*': c=a[0]*a[1]; S.push(c); break; case '+': c=a[0]+a[1]; S.push(c);</pre>			

```

STACK S;
int oper,a[2],c,cop,j;
int len;
char symbol;
cout<<"ENTER THE PREFIX EXPRESSION:";
gets(prefix);
len=strlen(prefix);

for(int i=len-1;i>=0;i--)
{
symbol=prefix[i];

```

```

break;
case '-': c=a[0]-a[1];
S.push(c);
break;
}
}
}
cout<<"RESULT : "<<S.pop();
}

```

+ A / * B % - C D E F

Token	Stack
F	F
E	F E
D	F E D
C	F E D C
-	F E (C-D)
%	F ((C-D)%E)
B	F ((C-D)%E) B
*	F (B* ((C-D)%E))
/	((B* ((C-D)%E))/F)
A	((B* ((C-D)%E))/F) A
+	(A+ ((B* ((C-D)%E))/F))
i=-1	(A+ ((B* ((C-D)%E))/F))

3 Write a user defined function to delete alternate characters from the given string. String can be a single word, or a sentence. **3**

Read string: 1 mark, Del: 2 marks. If students write only function full marks can be given. Deletion may be even positioned characters or odd positioned ones. The given function deletes all the even positioned characters.

```

class dchar
{
char str[50];
char key;
public:
void getstr()
{
cout<<"Enter the sentence:";
cin.get(str, 50, '\n');
}
}

```

```

void deletealternate()
{
int i=0, j=0;
while(i<strlen(str)){
str[j]=str[i+1];
i+=2;
j++;}
str[j]='\0';
cout<<str;
}
};

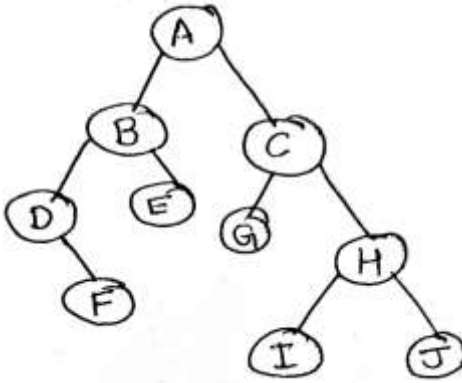
```

```

int main()
{
dchar d;
d.getstr();
d.deletealternate();
return 0;
}

```

4	<p>A doubly linked list is used to store ISSN, title, author, and publisher of many books. Write the class definition with suitable data members and member functions to perform the following operations on doubly linked list.</p> <p>i. Sort the list based on ISSN number [1.5m] ii. Delete all nodes of specific publisher. Specific publisher information is read from the user. [1.5m]</p> <pre> class Book { int ISSN; char title[20]; char author[20]; char publisher[10]; Book *prev; Book *next; public: Book(){} Book(int i,char t[],char a[],char p[]){ ISSN=i;strcpy(title,t); strcpy(author,a); strcpy(publisher,p); next=NULL; prev=NULL; } Book* sort(Book *); void swap(Book *,Book *); void delSpecific(); void create(int ,char [],char a[],char p[]); }; void Book::swap(Book *a,Book *b) { int ti;char t[10]; char au[10]; char pu[10]; ti=a->ISSN; strcpy(t,a->title);strcpy(au,a->author);strcpy(pu,a->publisher); a->ISSN=b->ISSN;strcpy(a->title,b->title);strcpy(a->author,b->author);strcpy(a->publisher,b->publisher); b->ISSN=ti; strcpy(b->title,t);strcpy(b->author,au);strcpy(b->publisher,pu); } Book* Book::sort(Book *f) { Book *t; for(Book *i=f;i->next!=NULL;i=i->next) { for(Book *j=i->next;j!=NULL;j=j->next) { if(i->ISSN>j->ISSN) { swap(i,j); } } } return(f); } void Book::delSpecific() { int x,flag=0; cout<<"\nenter the ISSN number to be deleted:"; cin>>x; if(first==NULL) cout<<"\nlist is empty"; else if(first->ISSN==x) </pre>	3
---	---	---

	<pre> { Book *temp=first; first=first->next; first->prev=NULL; delete temp; flag=1; } else { for(Book *curr=first;curr!=NULL;curr=curr->next){ if(curr->ISSN==x&&curr->next!=NULL) { curr->prev->next=curr->next; curr->next->prev=curr->prev; delete curr; flag=1; break; } else if(curr->ISSN==x) { curr->prev->next=NULL; delete curr; flag=1; break; } } } if(flag==0) cout<<"\nnode not found"; } </pre>	
5	<p>Write the inorder, postorder, preorder and level order traversal sequence for the tree shown in Figure 1. Write recursive user-defined functions to:</p> <p>a. Display the nodes of only the left subtree of the binary tree (1m)</p> <pre> void tree::inorderr(node *ptr) { static node *key=ptr; if(ptr) { inorderr(ptr->lchild); cout<<ptr->data<<" "; if (ptr==key) return; inorderr(ptr->rchild); } } </pre> <p>b. Count and return the total number of nodes with degree 2 (1m)</p> <p>[Inorder: 0.5m, postorder: 0.5m, preorder: 0.5m, level order: 0.5m]</p>	4
	 <pre> graph TD A((A)) --> B((B)) A --> C((C)) B --> D((D)) B --> E((E)) D --> F((F)) C --> G((G)) C --> H((H)) H --> I((I)) H --> J((J)) </pre> <p style="text-align: center;">Figure 1</p>	

	<pre> b. void tree::disp(node *r, int *co) { if(r) { disp(r->lchild,co); cout<<r->data; if(r->lchild!=NULL && r->rchild!=NULL) (*co)++; disp(r->rchild, co); } } </pre> <p>[inorder: 0.5m, preorder: 0.5m, postorder:0.5m, level order:0.5m] Inorder: DFBEAGCIHJ Postorder: FDEBGJIHCA Preorder: ABDFECGHIJ Level order: ABCDEGHFIJ</p>	
6	<p>(a) Write a user defined function to read a string from user, store each word of the input string as each node's value. For example, if the input string is "This is an example" then, first node stores the value "This," second node stores the value "is" and so on.</p> <p>(b) Write a user defined function to exchange the smallest and largest element in a singly linked list.</p> <p>((a) Read string: 0.5M; traverse and get a word:0.5; creating proper node: 0.5M; inserting suitably:0.5M (b) Identifying largest and smallest: 1.5M, swapping, 0.5M)</p>	4
	<pre> class STACK { int TOP; int a[30]; public: STACK(){TOP=-1;} void push(int); int pop(); }; void STACK::push(int ele) { if (TOP == 29) cout<<"stack is full; else a[++TOP]=ele; } int STACK::pop() { if(TOP== -1) return -999; else return a[TOP--]; } void prefix_evaluation() { STACK S; int oper,a[2],c,cop,j; int len; char symbol; cout<<"ENTER THE PREFIX EXPRESSION:"; gets(prefix); len=strlen(prefix); for(int i=len-1;i>=0;i--) { symbol=prefix[i]; </pre>	<pre> if(symbol >= 48 && symbol<= 57) S.push(symbol-'0'); else { for(j=0;j<2;j++) { a[j]=S.pop(); } switch(symbol) { case '%': c=a[0]%a[1]; S.push(c); break; case '/': c=a[0]/a[1]; S.push(c); break; case '*': c=a[0]*a[1]; S.push(c); break; case '+': c=a[0]+a[1]; S.push(c); break; case '-': c=a[0]-a[1]; S.push(c); break; } } cout<<"RESULT : "<<S.pop(); } </pre>