O –59411

18CSR234

Register No.

BE/BTech Degree Examination May 2019

Second Semester

Computer Science and Engineering

18CSC21 – PROGRAMMING AND LINEAR DATA STRUCTURES

(Regulations 2018)

Common to Information Technology

Time: Three hours Maximum: 100 marks

Answer all Questions

Part – A (10 × 2 = 20 marks)

1. Write any four benefits of pointers. [CO1,K1]

2. Predict the output for the following code snippet. [CO1,K2]

```
void main( )
{
int i = 3,*j,k;
j = &i;
printf(“%d”,i**j*i+*j);
}
```

3. Write a C program to find the multiplication of two numbers using function pointer. [CO2,K2]

4. Trace the output of the following code snippet [CO2,K2]

```
void fun (int*, int);
void main( )
{
  int i,a[ ]={1,2,3,4,5};
  fun(a,5);
  for(i = 0;i < 5;i ++)
  printf(“%d\t”,a[i]);
}
void fun(int *x,int n)
{
  int i;
 for i = 0;i < n;i ++)
* (x + 1) = *(x + i) + 10;
}
```

5. Compare text files and binary files. [CO3,K2]

6. Give the output for the following code snippet [CO3,K2]

```
#include<stdio.h>
#define a 10
int main( )
{
   printf(“%d”,a);
   #define a 50
   printf(“%d”,a);
   return 0;
}
```

7. Define a structure to create a node in a singly linked list. [CO4,K1]

8. Write a code snippet for inserting the element in the last position in a singly linked list. [CO4,K3]

9. If the elements ‘A’,’B’,’C’ and ‘D’ are placed in a queue and are deleted one at a time, specify the order in which they will be removed. [CO5,K2]

10. List any two applications of stack and queue. [CO5,K1]

Part – B (5 × 13 = 65 marks)

11. a. You are given the maths test mark for 'N' students of a class. Allocate memory dynamically for sorting the maths test mark of the students. (10) [CO1,K3]

Implement the following operations:

i) Find the number of students who have passed in the maths test.

ii) Find the number of students who have failed in the maths test

iii) Find the maximum mark in the maths test

iv) Count the number of students having their maths test mark above 80.

(OR)

b. i) Write a C program to manipulate the following string operations using pointers without using library functions (6) [CO1,K3]

1) Reverse the given string

2) Compare two strings

ii) Write a C program using pointers to find the range of the elements in a one dimensional array. (Note: Range of an array is the difference between the maximum and minimum element in an array). (6) [CO1,K3]

12. a. Develop two functions, first function is to find the maximum element in the matrix of order $n \times m$, and the second function is to find the sum of all the elements in the matrix of order $n \times m$.The functions should take matrix, row and column as an arguments and return the resultant value. Write a C program to perform these operations using function pointer. (12) [CO2,K3]

(OR)

b. A student result publishing system has to get the details of the students and then prints the result in rank order. Define a structure called STUDENT with data members Name, Roll No, three subject marks, total, average and rank. Perform the following operations by passing structure to a function by value. (12) [CO2,K3]

i) Get the student details for 'N' students

ii) Find the total, average of the students

iii) Find the rank of the students

iv) Display the details of the students in rank order

13. a. i) Write a C program to store the name and mobile number for 'N' users in a file named "input.txt". Perform the following operations: (8) [CO3,K3]

1) Read the contents from the file and display it on the monitor

2) Get an user name and display the corresponding mobile number. If the user name is not found, print the message, user name is not stored.

ii) Write a C program to find the sum of all the numbers given as command line arguments. (4) [CO3,K3]

(OR)

b. i) Develop a user defined header file named "calc.h" which contains user defined functions for performing simple arithmetic operations. Write a C program to include the header file "calc.h" to perform various arithmetic operations. (6) [CO3,K3]

ii) Write the difference between sequential access file and random access file and summarize the functions that are used to achieve random accessing in files with syntax and example program. (6) [CO3,K2]

14. a. You are given the height of each student of your class. Use singly linked list for performing the following operations. (12) [CO4,K3]

i) Storing the height of all the students using insert at beginning method.

ii) Find the height of the tallest student and delete from the list

iii) Display the heights and also count the number of students whose height is greater than 150 cm.

(OR)

b. Consider an XYZ organization needs to maintain the salary details of the employees. Develop a C program using singly linked list to implement the following operations: (12) [CO4,K3]

i) Read the salary of employee and store the salary in the beginning of the list.

ii) Get a salary 'k' of a new employee, and a search element 'x'. If 'x' is present in the list, insert 'k' after 'x', otherwise, print the message, "salary not found".

iii) If the salary of the employee of an organization is within the range:
15000 to 25000, provide a bonus of 10% of the salary,
25001 to 50,000 provide a bonus of 20% of the salary
50001 and above, provide a bonus of 30% of the salary
update the salary as salary = salary + bonus.

iv) Display the salary details present in the list and compute the total salary given by an organization for the employee.

15. a. Implement the operations of stack using linked list (12) [CO5,K3]

i) push( ) – store a number on to the stack

ii) pop( ) – delete a number from the stack

iii) display( ) – display all the numbers present in the stack.

iv) multiply( ) – If the number present in the stack is odd number, multiply it by 15. If it is even number, multiply it by 6.

(OR)

b. Write a C program to implement the following operations of queue using array. (12) [CO5,K3]

i) enqueue( ) – store a character on to the queue

ii) dequeue( ) – delete a character from the queue

iii) display( ) – display all the characters present in the queue

iv) count( ) – count the number of vowels, consonants, uppercase letters and lower case letters present in the queue.

Part – C (1 × 20 = 20 marks)

16. a. Define a structure called CRICKET with the data members playercode, playername, innings, notout, runs, bataverage and wickets. Implement the following operations by passing structure pointer as function arguments (20) [CO2,K3]

i) Get the input values for 'N' players

ii) Calculate the bataverage for each player, where bataverage = runs / innings–notout .

iii) Display the details of all the players

iv) Accept playercode as input and display the corresponding player details. If playercode does not exist, print the message "invalid player code".

(OR)

b\. ABC is an organization with creative and innovative ideas working towards making every event an unforgettable experience. The employees of the organization are allowed to register their employee-id (an integer value) in the arrival order and they are allowed to participate in the event as per the registration order. Identify the suitable data structure and write a C program to implement the various operations of the data structure using linked list. (20) [CO5,K3]

Bloom's Taxonomy Level	Remembering (K1)	Understanding (K2)	Applying (K3)	Analysing (K4)	Evaluating (K5)	Creating (K6)
Percentage	3.33	10	86.7	–	–	–

BE/BTech Examination-May 2019
18CSC21 – Programming and Linear Data Structures
Part - A

<table>
<tr><td>1</td><td>• Pointers provide direct access to memory
• Pointers are used to return more than one value from a function
• Pointers allow dynamic memory allocation and deallocation.
• Reduces the storage space required by the program
• Pointers are more efficient in handling arrays and structures
• Pointers are used to get reference of a variable or function.
• Pointers increases program execution speed
• Pointer allows to create complex data structures such as linked list, stack, queues, trees, graphs etc.</td><td>Any four points
2 mark</td></tr>
<tr><td>2</td><td>30</td><td>2 mark</td></tr>
<tr><td>3</td><td>#include<stdio.h>
int mul(int a, int b)
{
        return a*b;
}
void main()
{
        int c;
        int (*p)(int, int);
        p=&mul;
        c=(*p)(3,4);
        <b>(or)</b>
        p=mul;
        c=p(5,7);
        <b>(or)</b>
        int (*q)(int,int)=mul;
        c=q(4,4);
        printf("The value is %d \n", c);
}</td><td>1 mark
1 mark</td></tr>
<tr><td>4</td><td>1 15 3 4 5</td><td>2 mark</td></tr>
<tr><td>5</td><td><table>
<tr><th>Text File</th><th>Binary File</th></tr>
<tr><td>Text files contain ASCII codes of digits, alphabetic and symbols.</td><td>Binary files store data in the binary form (0's and 1's).</td></tr>
<tr><td>Extension of file is ".txt"</td><td>Extension of file is ".bin"</td></tr>
<tr><td>When opening text files, it is possible to see all the contents as plain text</td><td>When opening binary files, it is not readable easily by human as plain text</td></tr>
<tr><td>Provides least security</td><td>Provides better security than text file</td></tr>
</table></td><td>Any 2 points
2 mark</td></tr>
</table>

	Takes bigger storage space	They can hold higher amount of data with least storage space	
6	10 50		2 mark

7	struct node
	{
	    int data;
	    struct node *next;
	};
			2 mark

8	struct node
	{
	    int data;
	    struct node *next;
	};
	void insertionatend()
	{
	   int item;
	   struct node *newnode, *temp;
	   newnode=(struct*)malloc(sizeof(struct node));
	   newnode->data=item;
	   newnode->next=NULL;
	   temp=head;
	   while(temp->next!=NULL)
	   {
	      temp=temp->next;
	   }
	   temp->next = newnode;
	}
			2 mark

9	A->B->C->D		2 mark
10	**Stack**	**Queue**	Any 2 point from each 2 mark
	• Support of function calls • Support of recursive functions • An "undo/redo" mechanism in text editors • Expression Evaluation • Expression Conversion • Reverse a string • Forward and backward feature in web browsers • Parenthesis Checking	• CPU Scheduling • Printer Queue • Customer Service • Breadth First search in a Graph • Handling of interrupts in real-time systems. • Call Center phone systems	

Part – C (1 × 20 = 20 marks)

Define a structure called CRICKET with the data members playercode, (20) [CO2,K3]
runs bataverage and wickets. Implement the
arguments

PART - B

11.a	#include<stdlib.h>
	void main()
	{
	int n, *ptr, i, pass=0,fail=0,count=0,j,temp;
	printf("enter the number of students:");
	scanf("%d", &n);
	ptr = (int*) malloc (n*sizeof(int));
	if(ptr ==NULL)
	{
	printf("Memory not allocated");
	}
	else
	{
	printf("enter the maths mark for N students");
	for(i=0;i<n;i++)
	{
	scanf("%d",&ptr[i]);
	}
	}
	for(i=0;i<n;i++)
	{
	if(ptr[i]>=50)
	pass++;
	else
	fail++;
	}
	printf("Number of students who have passed in maths test:%d" , pass);
	printf("Number of students who have failed in maths test:%d", fail);
	for(i=0;i<n-1;i++)
	{
	for(j=i+1;j<n;j++)
	{
	if(ptr[i]>ptr[j])
	{
	temp= ptr[i];
	ptr[i]=ptr[j];
	ptr[j]=temp;
	}
	}
	}
	printf("Maximum mark in maths test:%d",ptr[n-1]);
	for(i=0;i<n;i++)
	{
	if(ptr[i]>80)
	count++;
	}
		2 mark
		2 mark
		4 mark
		2 mark
		2 mark
			12 mark

ayername, innings, notout, runs,

llowing operations by passing structure pointer as function argum

Get the input values for 'N' players runs /

	printf("Number of students having their maths test mark above 80:%d", count); }		
	(OR)		

11.b.i	#include<stdio.h>
	#include<conio.h>
	int stringcompare(char *, char *);
	char *stringreverse(char *);
	void main()
	{
	char a[10],b[10];
	int x;
	printf("enter string 1:");
	scanf("%s",a);
	printf("enter string 2:");
	scanf("%s",b);
	x=stringcompare(a,b);
	if(x==0)
	{
	printf(" Both string are equal");
	}
	if(x>0)
	{
	printf('string1 is greater than string2");
	}
	if(x<0)
	{
	printf("string2 is greater than string1");
	}
	printf("Reversed string:%s", stringreverse(a));
	}
	
	int stringcompare(char *a, char *b)
	{
	while(*a==*b)
	{
	if(*a=='\0' || *b=='\0')
	{
	break;
	}
	a++;
	b++;
	}
	if(*a == '\0' && *b=='\0')
	return 0;
	else
	return (*a-*b);
	}
		2 mark
		2 mark
			6 mark

	char *stringreverse(char *s)
	{
	static char rev[10];
	int l=0,x;
	l=strlen(s);
	x=l;
	while(*s)
	{
	rev[--l]=*s;
	s++;
	}
	rev[x]='\0';
	return(rev);
	}
		2 mark	

11.b.ii	#include<stdio.h>
	#include<conio.h>
	void main()
	{
	int a[10],n,i,j,temp;
	printf("enter the value of n:");
	scanf("%d",&n);
	printf("enter the elements of array");
	for(i=0;i<n;i++)
	{
	scanf("%d",(a+i));
	}
	for(i=0;i<n-1;i++)
	{
	for(j=i+1;j<n;j++)
	{
	if(*(a+i)>*(a+j))
	{
	temp=*(a+i);
	*(a+i)=*(a+j);
	*(a+j)=temp;
	}
	}
	}
	printf("Range of array:%d",*(a+n-1)-*(a+0));
	}
		2 mark
		3 mark
		1 mark
			6 mark

12.a.	#include <stdio.h>
	int add(int *a,int r,int c);
	int max (int *a,int r,int c);
	void main()
	{
	int a[3][3],r,c,i,j,res,res1;
	int (*addition)(int *a,int r,int c);
	int (*maximum)(int *a,int r,int c);
	printf("enter number of rows and columns:");
		2 mark	

	scanf("%d%d",&r,&c);
	printf("\nenter matrix elements:");
	for(i=0;i<r;i++)
	{
	for(j=0;j<c;j++)
	{
	scanf("%d",&a[i][j]);
	}
	}
		2 mark	

	addition = &add;
	res=(*addition)(a,r,c);
	printf("Sum of elements of matrix=%d",res);
		2 mark	

	maximum=&max;
	res1=(*maximum)(a,r,c);
	printf("Maximum element in matrix:%d",res1);
	}
		2 mark	

	int max(int *a,int r,int c)
	{
	int temp,i,j;
	temp=*(a);
	for(i=0;i<r;i++)
	{
	for(j=0;j<c;j++)
	{
	if(*(a+i*c+j)>temp)
	{
	temp=*(a+i*c+j);
	}
	}
	}
	return temp;
	}
		2 mark	12 mark

	int add(int *a,int r,int c)
	{
	int sum=0,i,j;
	for(i=0;i<r;i++)
	{
	for(j=0;j<c;j++)
	{
	printf("%d\t",*(a+i*c+j));
	sum=sum+*(a+i*c+j);
	}
	}
	return sum;
	}
		2 mark	

	(OR)		

12.b.	struct STUDENT
	{
	char name[25],rollno[20];
	int m1,m2,m3,total,rank;
	float average;
	}s[100];
		2 mark	12 mark

	void getdata(struct STUDENT s[],int n);
	void calculation(struct STUDENT s[],int n);
	void display(struct STUDENT s[],int n);
		2 mark	

	void main()
	{
	int n;
	printf("enter the number of students:");
	scanf("%d",&n);
	getdata(s,n);
	calculation(s,n);
	display(s,n);
	}
		2 mark	

	void getdata(struct STUDENT s[],int n)
	{
	int i;
	printf("enter the details of students");
	for(i=0;i<n;i++)
	{
	printf("Enter name:");
	scanf("%s",s[i].name);
	printf("Enter Rollno:");
	scanf("%s",s[i].rollno);
	printf("Enter M1,M2,M3:");
	scanf("%d%d%d",&s[i].m1,&s[i].m2,&s[i].m3);
	}
	}
		2 mark	

	void calculation(struct STUDENT s[],int n)
	{
	int i,j;
	struct STUDENT temp;
	for(i=0;i<n;i++)
	{
	s[i].total=s[i].m1+s[i].m2+s[i].m3;
	s[i].average=s[i].total/3;
	}
	for(i=0;i<n-1;i++)
	{
	for(j=i+1;j<n;j++)
	{
	if(s[i].total<s[j].total)
	{
		3 mark	

	temp=s[i];
	s[i]=s[j];
	s[j]=temp;
	}
	}
	}
	for(i=0;i<n;i++)
	{
	s[i].rank = i+1;
	}
	}
	
	void display(struct STUDENT s[],int n)
	{
	int i;
	printf("\n NAME \t ROLLNO\t M1\tM2\tM3\tTOTAL\t AVERAGE\tRANK\n");
	for(i=0;i<n;i++)
	{
	printf("%s\t%s\t%d\t%d\t%d\t%d\t%f\t%d\n",s[i].name, s[i].rollno,s[i].m1,s[i].m2,s[i].m3,s[i].total,s[i].average,s[i].rank);
	}
	}
		1 mark	

13.a.i	#include<stdio.h>
	void main()
	{
	FILE *fptr;
	char name[25],search[25];
	long int mobile;
	int n, flag=0;
	fptr=fopen("input.txt",'w+');
	printf("enter the N:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
	printf("enter name & Mobile number:");
	scanf("%s%ld",name,&mobile);
	fprintf(fptr,%s%ld",name,mobile);
	}
	rewind(fptr);
	while((fscanf(fptr,"%s%ld",name,&mobile))!=EOF)
	{
	printf("\nName:%s\tMobile:%ld",name,mobile);
	}
	rewind(fptr);
	printf("enter the name to search:");
	scanf("%s",search);
		1 mark
		1 mark
		2 mark
		2 mark
			8 mark

	while((fscanf(fptr,"%s%ld",name,&mobile))!=EOF)		
	{		
	if((strcmp(name,search)==0)		
	{		
	flag=1;		
	break;		
	}		
	}	2 mark	
	if(flag==1)		
	{		
	printf("Name found\nNAME:%s\tMOBILE:%ld", search,mobile);		
	}		
	else		
	printf("User name not stored");		
	}		
13.a.ii	void main(int argc, char *argv[])	1 mark	
	{		
	int i,sum=0;		
	if(argc>1)	1 mark	
	{		
	for(i=0; i<argc;i++)		
	{	1 mark	
	sum=sum+atoi(argv[i]);		4 mark
	}		
	}		
	else		
	printf("Insufficient Arguments");		
	printf("sum=%d",sum);	1 mark	
	}		
	(OR)		
13.b.i	**calc.h**		
	void add(int a,int b)		
	{		
	printf("sum=%d",a+b);		
	}		
	void sub(int a,int b)		
	{		
	printf("Difference=%d",a-b);		
	}		
	void mul(int a,int b)	3 mark	6 mark
	{		
	printf("Product=%d",a*b);		
	}		
	void div(int a,int b)		
	{		
	printf("Quo=%d",a/b);		
	}		

```
#include <stdio.h>
#include "calc.h"
void main()
{
    int a,b;
    printf("enter value of a and b:");
    scanf("%d%d",&a,&b);
    add(a,b);
    sub(a,b);
    mul(a,b);
    div(a,b);
}
```

1 mark

2 mark

13.b.ii

Sequential access File	Random access file
Information present in the file is accessed in a sequential fashion, one record after the other.	A random-access data file enables you to read or write information anywhere in the file.
It is faster when we need to access the information in the file always in the same order from first to last.	Slower
It is slower when we need to access information randomly.	Faster
Addition of record only at the end of the file is possible	New record can be added anywhere in the file

2 mark

To achieve random accessing in files, the following functions are used

- fseek()
- ftell()
- rewind()

2 mark

6 mark

fseek( )

Syntax: int fseek(FILE *fptr, long int Offset, int Position);

Example:

fseek(fp, 0, SEEK_END);

ftell()

Syntax: long ftell(FILE *pointer);

Example :

```
FILE *fp = fopen("test.txt","w");
char str[]="Hello World!;
 int p;
fprintf(fp,"%s",str); p=ftell(fp);
```

2 mark

rewind()

Syntax: void rewind(FILE *fptr);

Example:

```
    fptr=fopen("input.txt",'w+');
    printf("enter the N:");
```

```
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
    printf("enter name & Mobile number:");
    scanf("%s%ld",name,&mobile);
    fprintf(fptr,%s%ld",name,mobile);
 }
 rewind(fptr);
while((fscanf(fptr,"%s%ld",name,&mobile))!=EOF)
 {
    printf("\nName:%s\tMobile:%ld",name,mobile);
 }
```

14.a.

2 mark

```
#include<stdlib.h>
struct  node
{
      int data;
      struct node *next;
}*head=NULL;
void insertatbegin(int);
void delete();
void count();
void main()
{
   int n,i,height;
```

1 mark

```
   printf("enter the value of n:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("enter the height:");
      scanf("%d",&height);
      insertatbegin(height);
   }
   delete();
   count();
}

 void insertatbegin(int height)
 {
```

3 mark

```
      struct node *newnode;
      newnode=(struct node*)malloc(sizeof(struct node));
      newnode->data=height;
      newnode->next=head;
      head=newnode;
 }

void delete()
{
      int tallest,pos=1,del,i;
      struct node *temp,*todelete,*prev;
```

```
    tallest=head->data;

    temp=head;
    while(temp!=NULL)
    {
        pos++;
        if(temp->data>tallest)
        {
                tallest=temp->data;
                del=pos;
        }
        temp=temp->next;
    }
    prev=head;
    todelete=head;
    for(i=2;i<=del;i++)
    {
        prev=todelete;
        todelete=todelete->next;
        if(todelete==NULL)
            break;
    }
    if(todelete!=NULL)
    {
        if(todelete==head)
            head=head->next;
        prev->next=todelete->next;
        todelete->next=NULL;
    }
    free(todelete);
}

void count()
{
    int c=0;
    struct node *temp;
    temp=head;
    printf(“Heights of students are:”);
    while(temp!=NULL)
    {
        printf(“%d\t”,temp->data);
        if(temp->data >150)
        {
                c++;
        }
        temp=temp->next;
    }
    printf(“Number of students with height>150cm:%d”,c);
}
```

4 mark

12 mark

2 mark

(OR)

14.b

```
#include<stdlib.h>
struct  node
{
      float data;
      struct node *next;
}*head=NULL;
void insertatbegin(float);
void search(float);
void bonus();
void display();
void main()
{
   int n,i;
   float salary,nsalary;
   printf("enter the value of n:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("enter the height:");
      scanf("%f",&salary);
      insertatbegin(salary);
   }
   printf("enter the salary of new employee:");
   scanf("%f",&nsalary);
   search(nsalary);
   bonus();
   display();
}
```

2 mark

```
void insertatbegin(float salary)
 {
      struct node *newnode;
      newnode=(struct node*)malloc(sizeof(struct node));
      newnode->data=salary;
      newnode->next=head;
      head=newnode;
 }
```

2 mark

```
void search(float k)
{
     int pos=0,flag=0;
     float x;
     struct node *temp,*newnode,*l1;
     temp=head;
     printf("enter the search element:");
     scanf("%f",&x);
     while(temp!=NULL)
     {
        if(temp->data==x)
        {
           flag=1;
```

4 mark

	break;
	}
	pos++;
	temp=temp->next;
	}
	if(flag==0)
	{
	printf(“salary not found”);
	}
	else
	{
	
	printf(“salary found”);
	temp=head;
	newnode=(struct node *) malloc(sizeof(struct node));
	newnode->data=k;
	newnode->next=NULL;
	for(i=0;i<=pos-2;i++)
	{
	temp=temp->next;
	}
	l1=temp->next;
	temp->next=newnode;
	newnode->next=l1;
	}
	}
	
	void bonus()
	{
	struct node *temp;
	float b;
	temp=head;
	while(temp!=NULL)
	{
	if(temp->data>=15000 && temp->data<=25000)
	{
	b=temp->data*0.1;
	temp->data+=b;
	}
	if(temp->data>=25001 && temp->data<=50000)
	{
	b=temp->data*0.2;
	temp->data+=b;
	}
	if(temp->data>=50001)
	{
	b=temp->data*0.3;
	temp->data+=b;
	}
	temp=temp->next;
		2 mark	12 mark

```
}

void display()
{
    struct node *temp;
    float total=0.0;
    temp = head;
    while(temp!=NULL)
    {
          total+=temp->data
          printf(“%f”,temp->data);
          temp=temp->next;
     }
     printf(“Total Salary=%f”,total);
}
```

2 mark

15.a.

```
struct node
{
     int data;
     struct node *next;
}*top=NULL;
void push();
void pop();
void display();
void multiply();
void main()
{
     int choice;
     do
     {
          printf(“\n1.PUSH\n2.POP\n3.DISPLAY\n4.MULTIPLY
\n5.EXIT”);
          printf(“enter choice”);
          scanf(“%d”,&choice);
          switch(choice)
          {
               case 1: push(); break;
               case 2: pop(): break;
               case 3: display(); break;
               case 4 : multiply(); break;
               case 5: exit(0);
           }
      }while(choice!=4);
}

void push()
{
     int item;
     struct node *newnode;
     newnode  (struct node*)malloc(sizeof(struct node));
```

2 mark

2 mark

```
        printf("enter the element");
        scanf("%d",&item);
        newnode->data=item;
        newnode->next=top;
        top=newnode;
    }
```

2 mark

```
    void pop()
    {
        struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        if(top==NULL)
        {
            printf("\nStack Empty");
        }
        else
        {
            temp=top;
            printf("Popped element:%d",temp->data);
            top=temp->next;
            free(temp);
        }
    }
```

2 mark

12 mark

```
    void display()
    {
        printf("Elements are:");
        struct node *temp;
        temp=top;
        while(temp!=NULL)
        {
            printf("%d\t",temp->data);
            temp=temp->next;
        }
    }
```

2 mark

```
    void multiply()
    {
        struct node *temp;
        temp=top;
        while(temp!=NULL)
        {
            if(temp->data%2==0)
                temp->data*=6;
            else
                temp->data*=15;
            temp=temp->next;
        }
    }
```

2 mark

(OR)

15.b.	#include <stdio.h>
	#define size 5;
	void enqueue();
	void dequeue();
	void display();
	void count();
	int front=-1,rear=-1;
	char q[size];
	void main()
	{
	    int choice;
	    do
	    {
		2 mark	

	printf(“\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n4.COUNT \n5.EXIT”);
	        printf(“enter choice”);
	        scanf(“%d”,&choice);
	        switch(choice)
	        {
	            case 1: enqueue(); break;
	            case 2: dequeue(): break;
	            case 3: display(); break;
	            case 4 : count(); break;
	            case 5: exit(0);
	        }
	    }while(choice!=4);
	}
		2 mark	

	void enqueue()
	{
	    char elem;
	    ++rear;
	    if(rear>=size)
	    {
	        printf(“Queue Overflow”);
	    }
	    else
	    {
	        printf(“Enter the character”);
	        scanf(“%c”,&elem);
	        q[rear]=elem;
	    }
	}
		2 mark	

	void dequeue()
	{
	    if(front == rear)
	    {
	        printf(“Queue Underflow”);
	    }
		2 mark	

```
        else
        {
            front++;
            printf(“Dequeued element :%c”,q[front]);

            if(front ==rear)
            {
                front=rear= -1;
            }
        }
    }

    void display()
    {
        int i;
        if(front == rear)
        {
            printf(“queue empty”);
        }
        else
        {
            printf(“elements :”);
            for(i=front+1;i<=rear;i++)
            {
                printf(“%c”,q[i]);
            }
        }
    }

  void  count()
  {
    int v=0,c=0,u=0,l=0,i;
    for(i=front+1,i<=rear;i++)
    {
        if(isupper(q[i]))
            u++;
        if(islower(q[i]))
            l++;
        if(q[i]==’a’ || q[i]==’e’ || q[i]==’i’ || q[i]==’o’ || q[i]==’u’)
            v++;
        else
            c++;
    }
    printf(“vowels=%d\t consonants=%d\t uppercase letters=%d
\t lowercase letters=%d”,v,c,u,l);
}
```

2 mark

12 mark

2 mark

PART -C

16.a.	struct CRICKET
	{
	int playercode;
	char playername[25];
	int innings, notout, runs, wickets;
	float bataverage;
	}c[20];
		4 mark	20 mark

	void read(struct CRICKET c[],int n);
	void bataverages(struct CRICKET c[],int n);
	void display(struct CRICKET c[],int n);
	void check(struct CRICKET c[],int n);
		2 mark	

	void main()
	{
	int n,i;
	printf(“enter number of players:”);
	scanf(“%d”,&n);
	read(c,n);
	bataverages(c,n);
	display(c,n);
	check(c,n);
	}
		2 mark	

	void read(struct CRICKET c[],int n)
	{
	int i;
	printf(“enter players details”);
	for (i=0;i<n;i++)
	{
	printf(“enter player code:”);
	scanf(“%d”,&c[i].playercode);
	printf(“enter player name:”);
	scanf(“%s”,c[i].playername);
	printf(“enter out/notout:”);
	scanf(“%d”,&c[i].notout);
	printf(“enter runs:”);
	scanf(“%d”,&c[i].runs);
	printf(“enter innings:”);
	scanf(“%d”,&c[i].innings);
	printf(“enter wickets:”);
	scanf(‘%d”,&c[i].wickets);
	}
	}
		3 mark	

	void bataverages(struct CRICKET c[],int n)
	{
	int i;
	for(i=0;i<n;i++)		

3 mark

```
    {
        c[i].bataverage = c[i].runs/c[i].innings – c[i].notout;
    }
}
```

3 mark

```
void display(struct CRICKET c[],int n)
{
    int i;
    printf(“\nPlayercode\tPlayername\tInnings\tNotout\tRuns \t
Bataverage \t Wickets”);
     for(i=0;i<n;i++)
     {
          printf(“\n%d\t%s\t%d\t%d\t%d\t%f\t%d”,
c[i].playercode,c[i].playername,c[i].innings,c[i].notout, c[i].runs,
c[i].bataverage,c[i].wickets);
     }
}
```

3 mark

```
void check(struct CRICKET c[],int n)
{
    int x,flag=0;
    printf(“enter the playercode to check:”);
    scanf(‘%d”,&x);
    for(i=0;i<n;i++)
    {
        if(c[i].playercode==x)
        {
                flag=1;
                break;
        }
    }
    if(flag==1)
    {
          printf(“\nPlayercode\t Playername\t Innings\t Notout
\tRuns\t Bataverage \t Wickets”);
          printf(“\n%d\t%s\t%d\t%d\t%d\t%f\t%d”,
c[i].playercode, c[i].playername,c[i].innings,c[i].notout, c[i].runs,
c[i].bataverage,c[i].wickets);
     }
     else
          printf(“Invalid player code”);
}
```

(OR)

16.b.

2 mark

```
struct node
{
    int employee-id;
    struct node *next;
}*front=NULL,*rear=NULL;
void enqueue();
void dequeue();
void display();
void main()                                                        2 mark
{
   int choice;
   do
   {
        printf("\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n
4.EXIT");
        printf("enter choice");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: enqueue(); break;
            case 2: dequeue(): break;                              2 mark
            case 3: display(); break;
            case 4 : exit(0);
        }
      }while(choice!=4);
 }
 void enqueue()
 {
      struct node *newnode;
      int item;
      newnode=(struct node*)malloc(sizeof(struct node));
      printf("enter employee id");                                 5 mark
      scanf("%d",&item);
      newnode->employee-id= item;
      newnode->next=NULL;
      if(front ==NULL)
      {
           front=rear=newnode;
      }
      else
      {
           rear->next=newnode;
           rear=newnode;
       }
  }

  void dequeue()
  {
      if(front==NULL)
      {
           printf("Queue empty");
      }
      else
      {                                                                        20 mark
           struct node *temp;
           temp=front;                                             5 mark
           printf("Employee-id served:%d",temp->employee-id);
```

	front=front->next;
	free(temp);
	}
	}
	
	void display()
	{
	if(front==NULL)
	{
	printf(“queue empty”);
	}
	else
	{
	struct node *temp;
	temp=front;
	printf(“Employee-id to be served:”);
	while(temp!=NULL)
	{
	printf(“%d\t”,temp->employee-id);
	temp=temp->data;
	}
	}
	}
		4 mark	