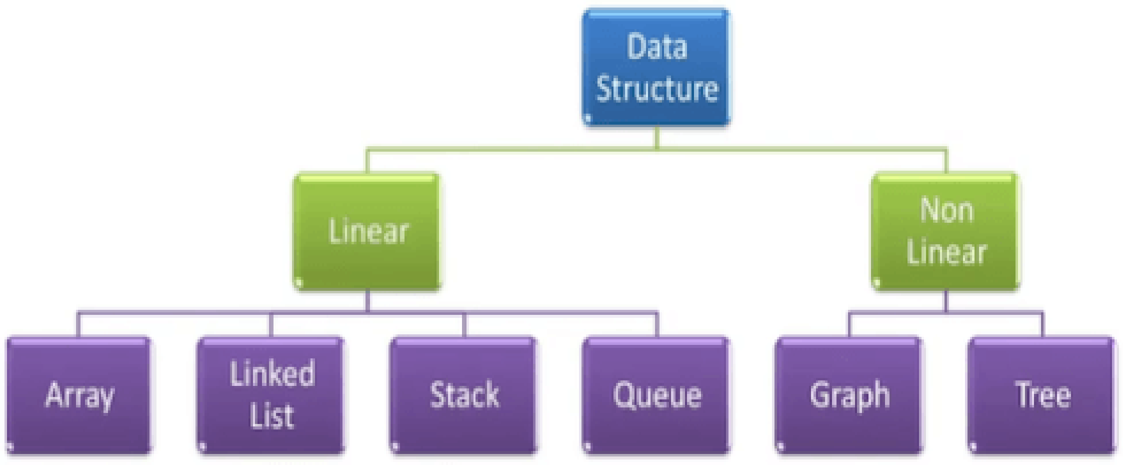
Department Of Computer Science and Engineering

Lab Manual (P18CSL37)



PES College of Engineering,

Mandya - 571401

DATA STRUCTURE LAB

Course Code: P18CSL37 **CIE: 50 Marks**
Credit Pattern: L:T:P = 0:0:1.5 = 1.5 **SEE: 50 Marks**
No. of Hours per Week: 3 **Exam Duration: 03 Hours**

Programs on Stacks

1. Write a C program to construct a stack and to perform the following operations.
   i) Push ii) Pop iii) Display
   The program should print appropriate message for stack overflow, stack underflow & stack empty.

2. Write a C program to convert and print a given valid parenthesized infix arithmetic expression to prefix expression. The expression consists of single character operands and binary operators + (Plus), - (Minus), * (Multiply), / (Divide).

3. Write a C program to evaluate a valid prefix expression using stack. Assume that the prefix expression is read as single line consisting of non-negative single digit operands and binary arithmetic operations.

4. Write a C program to check whether a given string is palindrome or not using stack.

Programs on Recursion

5. Write a recursive C program for
   i) To find larger of 'n' elements in an array
   ii) To multiply two natural numbers
   iii) Solving the Towers of Hanoi Problem

Programs on Queues

6. Write a C program to simulate the working of a queues using an array provide the following operation
   i) Insert ii) Delete iii) Display

7. Write a C program to simulate the working of a circular queues with items as strings. Provide the following operations
   i) Insert ii) Delete iii) Display

8. Write a C program to simulate the working of Double Ended Queue of integers using Structures. Provide the following operations
   i) Insert from front/rear end ii) Delete from front/rear end iii) Display

9. Write a C program to implement priority queues using structures (Assume a maximum of 3 queues).

Programs on Linked List

10. Write a C program using dynamic variables and pointers, to construct a Singly Linked List consisting of the following information in each node: Employee id (integer), Employee name (character string) and Department (character string). The operation to be supported are:
    a) The insertion operation
        i) At the front end of the list   ii) At the rear end of the list
        iii) At any position in the list

    b) Deleting a node based on employee id. If the specified node is not present in the list an error message should be displayed. Both the options should be demonstrated.

    c) Searching a node based on employee id and update node information. If the specified node is not present in the list an error message should be displayed. Both situations should be displayed.
    d) Displaying all the nodes in the list

11. Write a C program to construct a **Ordered Singly Linked List** and perform the following operations
    i) Reverse a list   ii) Concatenation of two lists

12. Write a C program to support the following operations on a Doubly Linked List where each node has the information as an integer
    i) Create a Doubly Linked List by adding each node at the front
    ii) Insert a new node to the right of the specified node.
    iii) To delete all nodes whose info is same as that of key item.
    iv) Display the contents of the list

Programs on Trees

13. Write a C program
    i) To create a Binary tree based on given direction.
    ii) To search for an item
    iii) To get the exact copy of a tree   iv) To display the elements based on its level from L-R

14. Write a C program
    i) To construct a BST of items (No Duplicates)
    ii) To search an item in BST
    iii) To display the elements of the tree using Inorder, preorder, & Postorder traversal methods.

Contents

Program 1:

Write a C program to construct a stack and to perform the following operations.
i) Push ii) Pop iii) Display
The program should print appropriate message for stack overflow, stack underflow & stack empty.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

#define MAX 5

/* Function to PUSH an element into Stack */
void push(int item,int *top,int S[])
{
    if(*top==MAX-1)
    {
    printf("stack overflow\n");
    return;
    }
    (*top)++;
    S[*top]=item;
    printf("%d is succesfully inserted\n", item);
}

/* Function to POP an element from the Stack */
void pop(int *top,int S[])
{
        int ele;
        if(*top==-1)
        {
                printf("stack is empty\n");
                return;
        }
        printf("%d element is deleted",S[*top]);
        (*top)--;
}

/* Function to display the elements of the Stack */
void display(int top,int S[])
{
        int i;
```

```
	if(top==-1)
	{
		printf("stack is empty\n");
		return 0;
	}
	printf("stack contains :\n");
	for (i=0;i<=top;i++)
	printf("%d\n", S[i]);
}


void main()
{
	int item,top=-1,S[MAX],choice;
	clrscr();
	for(;;)
	{
		printf("1.push \n  2.pop\n  3.display\n  4.exit\n");
		printf("\nEnter your choice:");
		scanf("%d",&choice);

		switch(choice)
		{
			case 1:printf("\nEnter an item:");
				scanf("%d",&item);
				push(item,&top,S);
				break;

			case 2:item=pop(&top,S);
				printf("%d element is deleted",item);
				break;

			case 3:display(top,S);
				break;

			default:printf("\ninvaild choice\n");
				 exit(0);
		}
	}
}
```

```
**********************************OUTPUT**********************************
1.push
2.pop
3.display
4.exit

Enter your choice:1

Enter an item:10
10 is succesfully inserted

1.push
2.pop
3.display
4.exit

Enter your choice:1

Enter an item:20
20 is succesfully inserted

1.push
2.pop
3.display
4.exit

Enter your choice:1

Enter an item:30
30 is succesfully inserted

1.push
2.pop
3.display
4.exit

Enter your choice:1

Enter an item:40
stack overflow

1.push
2.pop
3.display
```

```
4.exit

Enter your choice:3
stack contains :
10
20
30

1.push
2.pop
3.display
4.exit

Enter your choice:2
30 element is deleted
1.push
2.pop
3.display
4.exit

Enter your choice:2
20 element is deleted
1.push
2.pop
3.display
4.exit

Enter your choice:2
10 element is deleted
1.push
2.pop
3.display
4.exit

Enter your choice:2
stack is empty

1.push
2.pop
3.display
4.exit

Enter your choice:3
stack is empty
```

```
1.push
2.pop
3.display
4.exit

Enter your choice:
```

Program 2:

Write a C program to convert and print a given valid parenthesized infix arithmetic expression to prefix expression. The expression consists of single character operands and binary operators + (Plus), - (Minus), * (Multiply), / (Divide).

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>

#define max 20
char stack[max];
char temp;
int top=-1;

/* Function that returns Input Precedence values */
int prcd(char symb)
{
    switch(symb)
        {
                case '+':
                case '-':return 1;

                 case '*':
                case '%':
                case '/':return 3;

                case '^':
                case '$':return 6;

                case '#':return -1;

                case ')':return 0;

                 default:return 8;
        }
}

/* Function that returns Stack Precedence values */
int isop(char symb)
{
        switch (symb)
        {
                case '+':
                case '-':return 2;

                case '*':
                case '%':
                case '/':return 4;

                case '$':
                case '^':return 5;

                case '(':return 0;

                case ')':return 9;

                default:return 7;
        }
}
```

/* Function to convert an infix expression to postfix expression */

```
void infix_prefix(char inf[],char prf[])
{
        int i,j=0;
        char symb;

        stack[++top]='#';

        for(i=strlen(inf)-1;i>=0;i--)
        {
                symb=inf[i];
                while(prcd(stack[top])>isop(symb))
                {
                        prf[j++]=stack[top--];
                }
                if(prcd(stack[top])!=isop(symb))
                        stack[++top]=symb;
                else
                         top--;
        }

        while(stack[top]!='#')
       {
                prf[j]=stack[top];
                 j++;
                top--;
```

```
            }

            prf[j]='\0';

            printf("Expression in Prefix Format: ");
             for(i=strlen(prf)-1;i>=0;i--)
                        printf("%c",prf[i]);
}




void main()
{
 char inf[30],prf[30];
 printf("Enter an Expression in Infix format:\n");
 scanf("%s",inf);
 infix_prefix(inf,prf);
}
```

**********************************OUTPUT**********************************

```
Enter an Expression in Infix format:
(A+(B-C)*D)
Expression in Prefix Format: +A*-BCD
```

Program 3:

Write a C program to evaluate a valid prefix expression. Assume that the prefix expression has single digit operands and binary arithmetic operations.

```
#include<stdio.h>
#include<math.h>
#include<ctype.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 50
int S[MAX], top=-1;

/* Function to PUSH an element */
void push(int item)
{
        S[++top]=item;
}

/* Function to POP an element */
int pop()
{
        int ele;
        ele=S[top--];
        return ele;
}

void main()
{
        char prfx[50],ch;
        int i=0,a,b;
        clrscr();
        printf("enter a prefix expression\n");
        scanf("%s",prfx);
        strrev(prfx);
        printf("\n%s",prfx);
        while((ch=prfx[i++])!='\0')
        {
                if(isdigit(ch))
                        push(ch-'0');
                else
                {
                        a=pop();
```

```
                              b=pop();
                              switch(ch)
                              {
                                         case'+': push(a+b);
                                                    break;
                                        case'-': push(a-b);
                                                    break;
                                        case'*': push(a*b);
                                                    break;
                                        case'/': push(a/b);
                                                    break;
                                        case'^': push(pow(a,b));
                                                    break;
                              }
                    }
          }
          printf("\nResult is: %d", S[top]);
}
```

```
**************************************OUTPUT*******************************

enter a prefix expression
+9*-422

Result is: 5
```

Program 4:

Write a C program to check whether a given string is palindrome or not using stack.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define MAX 25
char ST[MAX];

void push(int item,int *top,char ST[])
{
        (*top)++;
        ST[*top]=item;
}

void main()
{
        char s[50];
         int top=-1,bot,len,flag,i;
        clrscr();
        printf("Enter the string:");
        scanf("%s", s);
        for(i=0;s[i]!='\0';i++)
                push(s[i],&top,ST);

         bot=0;
        while(bot<=top)
        {
                   if(s[top]==s[bot])
                {
                           top=top-1;
                          bot=bot+1;
                           flag=1;
                }
                else
                {
                        flag=0;
                        break;
                }
        }
        if(flag==1)
                printf("%s is a palindrome",s);
```

```
	else
		printf("%s is not a palindrome",s);
	getch();
}
```

```
**********************************OUTPUT******************************

Enter the string: madam
madam is a palindrome

Enter the string: madan
madan is not a palindrome
```

Program 5:

Write a recursive C program for

- **i) To find larger of 'n' elements in an array**
- ii) **To multiply two natural numbers**
- iii) **Solving the Towers of Hanoi Problem**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

/* Function to find maximum amongst n elements */
int findMaxElement(int arr[],int size)
{
        int static i=0, max=-999;
        if(i<size)
        {
                if(max<arr[i])
                {
                        max=arr[i];
                }
                i++;
                findMaxElement(arr,size);
        }
        return max;
}

/* Function to find product of two natural numbers */
int product(int a,int b)
{
        if(a<b)
        {
                return product(b,a);
        }
        else if(b!=0)
        {
                return (a+product(a,b-1));
        }
        else
                return 0;
}
```

```
/* Function definition for Tower_of_hanoi */
void tower(int n,char src,char aux,char dest)
{
        if(n==1)
        {
                printf("Move disk %d from peg %c to peg %c",n,src,dest);
                return;
        }
        tower((n-1),src,dest,aux);
        printf("\nMove disk %d from peg %c to peg %c\n",n,src,dest);
        tower((n-1),aux,src,dest);
}

void main()
{
        int arr[20];
        int i,n,max;
        int a,b,res;
        int num,ch;
        //clrscr();

        for(;;)
        {
                printf(" 1 : find Max Element\n");
                printf(" 2 : Multiplication of two natural numbers\n");
                printf(" 3 : Tower of Hanoi\n");
                printf("Any other choice : Exit\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: printf("Enter the size of the array\n");
                                scanf("%d",&n);
                                printf("Enter %d elements of an array\n",n);
                                for(i=0;i<n;i++)
                                {
                                        scanf("%d",&arr[i]);
                                }
                                max=findMaxElement(arr,n);
                                printf("Maximum element in the array is %d\n",max);
                                break;
                        case 2: printf("Enter two numbers to find their product:\n");
                                scanf("%d%d",&a,&b);
                                res=product(a,b);
```

```
                                        printf("Product of %d and %d is %d\n",a,b,res);
                                        break;
                              case 3: printf("Enter the number of disks :\n");
                                        scanf("%d",&num);
                                        printf("The sequence of moves involved in the Tower of Hanoi are
                                         :\n");
                                        tower(num,'A','C','B');
                                        break;
                              default:exit(0);
                    }
          }
}
```

```
***********************************OUTPUT****************************
1 : find Max Element
2 : Multiplication of two natural numbers
3 : Tower of Hanoi
Any other choice : Exit
Enter your choice
1
Enter the size of the array
5
Enter 5 elements of an array
20 30 50 25 15
Maximum element in the array is 50

 1 : find Max Element
 2 : Multiplication of two natural numbers
 3 : Tower of Hanoi
Any other choice : Exit
Enter your choice
2
Enter two numbers to find their product:
99 10
Product of 99 and 10 is 990

 1 : find Max Element
 2 : Multiplication of two natural numbers
 3 : Tower of Hanoi
Any other choice : Exit
Enter your choice
3
Enter the number of disks :
3
```

```
The sequence of moves involved in the Tower of Hanoi are :
Move disk 1 from peg A to peg B
Move disk 2 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 3 from peg A to peg B
Move disk 1 from peg C to peg A
Move disk 2 from peg C to peg B
Move disk 1 from peg A to peg B

1 : find Max Element
 2 : Multiplication of two natural numbers
 3 : Tower of Hanoi
Any other choice : Exit
Enter your choice
4
```

Program 6:

Write a c program to stimulate the working of a queues using an array provided the following operations

1. insert 2. delete 3. display

```
 #include<stdio.h>
 #include<conio.h>
 #include<stdlib.h>
 #define MAX 3
 int q[MAX], f=0, r=-1;

/* Function to insert an element into queue */
void insertion(int item)
 {
            if(r==MAX-1)
            {
                        printf("queue is full\n");
                        return;
            }
            q[++r]=item;
            printf("%d is inserted",item);
 }

/* Function to delete an element from the queue */
void deletion()
 {
             int ele;
             if(f>r)
             {
                        printf("queue is empty\n");
                        f=0;
                         r=-1;
                         return;
             }
             ele=q[f++];
             printf("%d is deleted", ele);
 }


/* Function to display the elements of queue */
 void display()
 {
             int i;
             if(f<=r)
            {
                        printf("queue contains:\n");
                        for(i=f; i<=r; i++)
                        printf("%d\n", q[i]);
             }
            else
                         printf("queue is empty");
 }


 void main()
 {
            int item, ch;
            clrscr();
            printf("welcome\n");
            while(1)
            {
                         printf("\n1.insert\n2.delete\n3.dispay\n4.quit\n");
                        printf("\nEnter your choice\n");
                        scanf("%d", &ch);
                        switch(ch)
                        {
                                    case 1: printf("\nEnter a item\n");
                                            scanf("%d", &item);
                                            insertion(item);
                                            break;
                                    case 2: deletion();
                                            break;
                                    case 3: display();
                                            break;
                                    default:printf("invalid choice");
                                            exit(0);
                        }
            }
 }
```

```
************************************OUTPUT*********************************
welcome

1.insert
2.delete
3.dispay
4.quit
```

```
Enter your choice
1

Enter a item
100
100 is inserted
1.insert
2.delete
3.dispay
4.quit

Enter your choice
1

Enter a item
200
200 is inserted
1.insert
2.delete
3.dispay
4.quit

Enter your choice
1

Enter a item
300
300 is inserted
1.insert
2.delete
3.dispay
4.quit

Enter your choice
1

Enter a item
400
queue is full

1.insert
2.delete
3.dispay
```

```
4.quit

Enter your choice
3
queue contains:
100
200
300

1.insert
2.delete
3.dispay
4.quit

Enter your choice
2
100 is deleted
1.insert
2.delete
3.dispay
4.quit

Enter your choice
2
200 is deleted
1.insert
2.delete
3.dispay
4.quit

Enter your choice
2
300 is deleted
1.insert
2.delete
3.dispay
4.quit

Enter your choice
2
queue is empty

1.insert
2.delete
```

```
3.dispay
4.quit

Enter your choice
3
queue is empty
1.insert
2.delete
3.dispay
4.quit

Enter your choice
```

Program 7:

Write a C program to simulate the working of a circular queues with items as strings. Provide the following operations

i) Insert ii) Delete iii) Display

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define max 3
char q[max][20];
int r=-1,f=0;
int c=0;

/* Function to insert an element into circular queue */
void insert()
{
        char item[10];
        if(c==max)
        {
                printf("Queue is full\n");
        }
        else
        {
                printf("Enter the string to be inserted = \n");
                r=(r+1)%max;
                scanf("%s",item);
                strcpy(q[r],item);
                c++;
        }
}

/* Function to delete an element from circular queue */
void del()
{
        if(c==0)
        {
                printf("Queue is empty\n");
        }
        else
        {
                printf("The deleted element is ");
                puts(q[f]);
                f=(f+1)%max;
                c--;
        }
}


/* Function to display the elements of circular queue */
void display()
{
        int i,j;
        if(c==0)
        {
                printf("Queue is Empty\n");
        }
        else
        {
                j=f;
                printf("The contents of the queue are");
                for(i=0;i<c;i++)
                {
                        printf(" %s\n",q[j]);
                        j=(j+1)%max;
                }
                printf("\n");
        }
}

int main()
{
        int ch;
        while(1)
        {
                printf("Choice 1 : Queue_Insert\n");
                printf("Choice 2 : Queue_Delete\n");
                printf("Choice 3 : Queue_Display\n");
                printf("Any other choice : Exit\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1 : insert();
                                break;
                        case 2: del();
                                break;

                        case 3: display();
                                break;

                        default: exit(0);
                }
        }
        return 0;
}
```

```
******************************OUTPUT**********************************
Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
1
Enter the string to be inserted:
aaa
Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
1
Enter the string to be inserted:
bbb
Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
1
Enter the string to be inserted:
ccc
Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
1
Queue is full
```

```
Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
3
The contents of the queue are
 aaa
 bbb
 ccc

Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
2
The deleted element is: aaa
Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
3
The contents of the queue are
 bbb
 ccc

Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
1
Enter the string to be inserted:
zzz
Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
3
The contents of the queue are
 bbb
 ccc
 zzz

Choice 1 : Queue_Insert
Choice 2 : Queue_Delete
Choice 3 : Queue_Display
Any other choice : Exit
Enter your choice
4
```

Program 8:

Write a C program to simulate the working of Double Ended Queue of integers using Structures. Provide the following operations
i) Insert from front/rear end ii) Delete from front/rear end iii) Display
.

```
  #include<stdio.h>
  #include<stdlib.h>
  #include<conio.h>
  #define QS 4

/*structure declaration for double ended queue*/
 struct queue
  {
    int q[QS];
    int r;
    int f;
  };

/*function prototypes*/
  void insert_rear(int ele, struct queue*);
  void insert_front(int ele, struct queue*);
  void delete_rear(struct queue*);
  void delete_front(struct queue*);
  void display(struct queue);


  void main()
  {
          int ch, ele;
          struct queue q1;
          clrscr();
          q1.r=-1;
          q1.f=0;
          while(1)
          {
                  printf("\n1.insert_rear\n2.insert_front\n3.delete_rear\n4.delete_front\n5.display\n
                  6.quit\n");
                  printf("\nEnter your choice\n");
                  scanf("%d", &ch);
                  switch (ch)
                  {
                          case 1:printf("\nEnter an element\n");
```

```
                         scanf("%d",  &ele);
                        insert_rear(ele, &q1);
                        break;

                case 2:printf("\nEnter an element\n");
                        scanf("%d", &ele);
                        insert_front(ele, &q1);
                        break;

                case 3:delete_rear(&q1);
                         break;

                 case 4:delete_front(&q1);
                         break;

                 case 5:display(q1);
                         break;

                default: printf("invalid choice\n");
                        exit(0);
            }
        }
  }
```

/* Function to insert an element at the rear end of the double ended queue */

```
void insert_rear(int ele, struct queue *q1)
  {
        if(q1->r==QS-1)
        {
                printf("queue is full\n");
                 return;
         }
        (q1->r)++;
         q1->q[q1->r]=ele;
        printf("%d is inserted\n", ele);
  }
```

/* Function to insert an element at the front end of the double ended queue */

```
void insert_front(int ele, struct queue *q1)
  {
        if(q1->f==0 && q1->r==-1)
        {
                q1->q[q1->f]=ele;
                return;
```

```
            }
           if(q1->f!=0)
           {
                       (q1->f)--;
                       q1->q[q1->f]=ele;
                       printf("%d is inserted\n",ele);
                       return;
           }
           printf("insertion not possible\n");
   }
```

/* Function to delete an element from the rear end of the double ended queue */

```
void delete_rear(struct queue *q1)
{
           if(q1->f > q1->r)
           {
                       printf("queue is empty\n");
                       q1->f=0, q1->r=-1;
                       return;
           }
           printf("%d is deleted",q1->q[q1->r]);
           (q1->r)--;
}
```

/* Function to delete an element from the front end of the double ended queue */

```
void delete_front(struct queue *q1)
 {
            if(q1->f > q1->r)
            {
                       printf("queue is empty\n");
                       q1->f=0, q1->r=-1;
                        return;
           }
           printf("%d is deleted", q1->q[q1->f]);
           (q1->f)++;
 }
```

/* Function to display the elements of the double ended queue */

```
void display(struct queue q1)
  {
            int i;
            if(q1.f<=q1.r)
            {
```

```
                        printf("queue contains:\n");
                        for(i=q1.f; i<=q1.r; i++)
                                printf("%d\n", q1.q[i]);
            }
            else
                         printf("queue is empty\n");
 }
```

```
************************************OUTPUT*******************************
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
1

Enter an element
10
10 is inserted

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
1

Enter an element
20
20 is inserted

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit
```

```
Enter your choice
1

Enter an element
30
30 is inserted

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
5
queue contains:
10
20
30

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
2

Enter an element
50
insertion not possible

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
```

```
4
10 is deleted
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
5
queue contains:
20
30

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
2

Enter an element
100
100 is inserted

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
5
queue contains:
100
20
30

1.insert_rear
```

```
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
3
30 is deleted
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
5
queue contains:
100
20

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
4
100 is deleted
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
5
queue contains:
20
```

```
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
3
20 is deleted
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
4
queue is empty

1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.quit

Enter your choice
```

Program 9:

Write a c program to implement priority queues using structures (assume a maximum of 3 queues)

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 3

/*Structure declaration for queues*/
struct queue
{
   int q[10];
   int r;
   int f;
};

/* Function to insert an element at the rear end of the queue */
void insert_rear(int item, struct queue *Q)
{
        if(Q->r==MAX-1)
        {
                printf("queue is full\n");
                return;
        }
        Q->r++;
        Q->q[Q->r]=item;
        printf("%d is inserted\n",item);
}

/* Function to delete an element from the front end of the queue */
int delete_front(struct queue *Q)
{
        int ele;
        if(Q->f > Q->r)
        {
                Q->f=0, Q->r=-1;
                return -1;
        }
        ele=Q->q[Q->f++];
        printf("%d is deleted\n", ele);
        return ele;
}
```

/* Function to display the elements of the queue */

```
void display(struct queue Q)
{
        int i;
        if(Q.f<=Q.r)
        {
                for(i=Q.f; i<=Q.r; i++)
                printf("%d\n",Q.q[i]);
        }
        else
                printf("queue is empty\n");
}

void main()
{
        int item, ch, pr;
        struct queue q1, q2, q3;
        q1.r=q2.r=q3.r=-1;
        q1.f=q2.f=q3.f=0;
        while(1)
        {
                printf("\n1.insertion\n  2.deletion\n  3.display\n");
                printf("\nEnter a choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("\nEnter an item");
                                scanf("%d", &item);
                                printf("\nEnter the priority");
                                scanf("%d", &pr);
                                switch(pr)
                                {
                                        case 1:insert_rear(item, &q1);
                                                break;
                                        case 2:insert_rear(item, &q2);
                                                break;
                                        case 3:insert_rear(item, &q3);
                                                break;
                                }
                                break;

                        case 2:if(delete_front(&q1)==-1)
                                {
                                        printf("queue 1 is empty\n");
                                        if(delete_front(&q2)==-1)
                                        {
                                                printf("queue 2 is empty\n");
                                                if(delete_front(&q3)==-1)
                                                        printf("queue 3 is empty\n");
                                        }
                                }
                                break;

                        case 3: printf("Contents of queue 1 is:\n");
                                display(q1);

                                printf("Contents of queue 2 is:\n");
                                display(q2);

                                printf("Contents of queue 3 is:\n");
                                display(q3);

                                break;

                        default: exit(0);
                }
        }
}
```

```
***************************************OUTPUT******************************
1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 1

Enter the priority: 1
1 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1
```

```
Enter an item: 2

Enter the priority: 1
2 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 3

Enter the priority: 1
3 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 10

Enter the priority: 2
10 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 20

Enter the priority: 2
20 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 30

Enter the priority: 2
30 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 100

Enter the priority: 3
100 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 200

Enter the priority: 3
200 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
1

Enter an item: 300

Enter the priority: 3
```

```
300 is inserted

1.insertion
  2.deletion
  3.display

Enter a choice
3
Contents of queue 1 is:
1
2
3
Contents of queue 2 is:
10
20
30
Contents of queue 3 is:
100
200
300

1.insertion
  2.deletion
  3.display

Enter a choice
2
1 is deleted

1.insertion
  2.deletion
  3.display

Enter a choice
2
2 is deleted

1.insertion
  2.deletion
  3.display

Enter a choice
2
3 is deleted
```

```
1.insertion
  2.deletion
  3.display

Enter a choice
2
queue 1 is empty
10 is deleted

1.insertion
  2.deletion
  3.display

Enter a choice
2
queue 1 is empty
20 is deleted

1.insertion
  2.deletion
  3.display

Enter a choice
3
Contents of queue 1 is:
queue is empty
Contents of queue 2 is:
30
Contents of queue 3 is:
100
200
300

1.insertion
  2.deletion
  3.display

Enter a choice
2
queue 1 is empty
30 is deleted

1.insertion
```

```
  2.deletion
  3.display

Enter a choice
2
queue 1 is empty
queue 2 is empty
100 is deleted

1.insertion
  2.deletion
  3.display

Enter a choice
3
Contents of queue 1 is:
queue is empty
Contents of queue 2 is:
queue is empty
Contents of queue 3 is:
200
300

1.insertion
  2.deletion
  3.display

Enter a choice
4
```

Program 10:

Write a C program using dynamic variables and pointers, to construct a Singly Linked List consisting of the following information in each node: Employee id (integer), Employee name (character string) and Department (character string). The operation to be supported are:

a) **The insertion operation**
   i) At the front end of the list ii) At the rear end of the list iii) At any position in the list

b) **Deleting a node based on employee id. If the specified node is not present in the list an error message should be displayed. Both the options should be demonstrated.**

c) **Searching a node based on employee id and update node information. If the specified node is not present in the list an error message should be displayed. Both situations should be displayed.**

d) **Displaying all the nodes in the list**

```
#include<stdio.h>
#include<alloc.h>
#include<process.h>
#include<string.h>

/*structure declaration for the employee*/
struct employee
{
        char name[10];
        int id;
        char dept[10];
        struct employee* link;
};
typedef struct employee *EMP;

/*function to allocate memory */
EMP getnode()
{
        EMP x;
        x=(EMP)malloc(sizeof(struct employee));
        if(x==NULL)
        {
                printf("Out of memory\n");
                exit(0);
        }
        return x;
}

/*function to insert a node at the front end of the list*/
EMP insert_front(char name[],int id,char dept[],EMP first)
{
        EMP temp;
        temp=getnode();
        strcpy(temp->name,name);
        strcpy(temp->dept,dept);
        temp->id=id;
        temp->link=first;
        first=temp;
        return first;
}

/*function to insert a node at the rear end of the list*/
EMP insert_rear(char name[],int id,char dept[],EMP first)
{
        EMP temp;
        EMP cur;
        temp=getnode();
        strcpy(temp->name,name);
        temp->id=id;
        strcpy(temp->dept,dept);
        temp->link=NULL;
        if(first==NULL)
        return temp;
        cur=first;
        while(cur->link!=NULL)
        {
                cur=cur->link;
        }
        cur->link=temp;
        return first;
}


/*function to insert a node at a specified position in the list*/
EMP insert_pos(char name[],int id,char dept[],int pos,EMP first)
{
        EMP temp;
        EMP cur,prev;
        int count;
        temp=getnode();
        strcpy(temp->name,name);
        temp->id=id;
        strcpy(temp->dept,dept);
        temp->link=NULL;
        if(first==NULL&&pos==1)
                return temp;
        if(first==NULL)
        {
                printf("Invalid position\n");
                return first;
        }
        if(pos==1)
        {
                temp->link=first;
                return temp;
        }
        count=1;
        prev=NULL;
        cur=first;
        while(cur!=NULL && count !=pos)
        {
                prev=cur;
                cur=cur->link;
                count++;
        }
        if(count==pos)
        {
                prev->link=temp;
                temp->link=cur;
                return first;
        }
        printf("Invalid position\n");
        return first;
}
```

/*function to delete a node from the list based on id*/

```
EMP delete_employee(int id,EMP first)
{
        EMP prev,cur,temp;
        if(first==NULL)
        {
                printf("No employees in the organisation\n");
                return NULL;
        }
      if(id==first->id)
        {
                temp=first;
           first=first->link;
                free(temp);
                return first;
        }
        prev=NULL;
        cur=first;
        while(cur!=NULL && id!=cur->id)
        {
                prev=cur;
                cur=cur->link;
        }
        if(cur==NULL)
        {
                printf("Employee id not found\n");
                return first;
        }


        prev->link=cur->link;
        free(cur);
        return first;
}
```

/*function to search for a node in the list based on id*/

```
EMP search(int id,EMP first)
{
        EMP cur;
        char name[10],dept[10];
        if(first==NULL)
        {
                printf("No employees in the organisation\n");
                return first;
        }
        cur=first;
        while(cur!=NULL&&id!=cur->id)
        {
                cur=cur->link;
        }
        if(cur==NULL)
```

```
	{
		printf("Employees with id %d not found\n",id);
		return first;
	}
	printf("Employee found with following information\n");
	printf("Name:%s\nID:%d\nDept:%s\n",cur->name,cur->id,cur->dept);
	printf("Enter the Name and Dept to update\n");
	scanf("%s%s",name,dept);
	strcpy(cur->name,name);
	strcpy(cur->dept,dept);
	return first;
}

/*function to display the details of employees in the list*/
void display(EMP first)
{
	EMP temp;
	if(first==NULL)
	{
		printf("No employees in the organisation\n");
		return;
	}
	printf("Employee name\tEmployee ID\tDepartment\n");
	printf("------------------------------------------\n");
	for(temp=first;temp!=NULL;temp=temp->link)
		printf("%3s\t\t%3d\t\t%3s\n",temp->name,temp->id,temp->dept);
	printf("\n");
}

void main()
{
	EMP first=NULL;
	int choice,id,pos;
	char name[10],dept[10];
	clrscr();
	for(;;)
	{
		printf("\n1:Insert_front\n2:Insert_rear\n");
		printf("3:Insert at position\n4:delete\n");
		printf("5:search\n6:display\n");
		printf("7:exit\n");
		printf("Enter the choice\n");
		scanf("%d",&choice);
		if(choice==1||choice==2||choice==3)
```

```
                {
                        printf("Name:");scanf("%s",name);
                        printf("ID:");scanf("%d",&id);
                        printf("Dept:");scanf("%s",dept);
                }
                switch(choice)
                {
                        case 1:first=insert_front(name,id,dept,first);
                                break;

                        case 2:first=insert_rear(name,id,dept,first);
                                break;

                        case 3:printf("Enter position to insert\n");
                                scanf("%d",&pos);
                                first=insert_pos(name,id,dept,pos,first);
                                break;

                        case 4:printf("Delete employee details for Id:\n");
                                scanf("%d",&id);
                                first=delete_employee(id,first);
                                break;

                        case 5:printf("Search with Id:");
                                scanf("%d",&id);
                                first=search(id,first);
                                break;

                        case 6:display(first);
                                break;

                        default:printf("Wrong choice...\n");
                                 break;
                }
        }
}
```

************************************OUTPUT******************************

```
1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
```

```
6: display
7: exit
Enter the choice
1
Name:aaa
ID:1
Dept:cse

1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
1
Name:bbb
ID:2
Dept:ece

1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
2
Name:ccc
ID:3
Dept:mech

1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
6
Employee name        Employee ID   Department
```

```
------------------------------------------
bbb               2               ece
aaa               1               cse
ccc               3               mech


1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
3
Name:zzzz
ID:2    4
Dept:civil
Enter position to insert
2

1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
6
Employee name       Employee ID  Department
------------------------------------------
bbb               2               ece
zzz               4               civil
aaa               1               cse
ccc               3               mech


1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
```

```
7: exit
Enter the choice
5
Search with Id:4
Employee found with following information
Name: zzz
ID:4
Dept: civil
Enter the Name and Dept to update
xyz
ise

1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
6
Employee name       Employee ID   Department
------------------------------------------
bbb                 2               ece
xyz                 4               ise
aaa                 1               cse
ccc                 3               mech


1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
4
Delete employee details for Id:
3

1: Insert_front
2: Insert_rear
3: Insert at position
```

```
4: delete
5: search
6: display
7: exit
Enter the choice
6
Employee name         Employee ID   Department
-------------------------------------------
bbb              2              ece
xyz              4              ise
aaa              1              cse


1: Insert_front
2: Insert_rear
3: Insert at position
4: delete
5: search
6: display
7: exit
Enter the choice
```

Program 11:

Write a C program to construct a Ordered Singly Linked List and perform the following operations

i) Reverse a list **ii) Concatenation of two lists**

```
#include<stdio.h>
//#include<alloc.h>
//#include<process.h>
#include<stdlib.h>

typedef struct node
{
        int info;
        struct node *link;
}*NODE;


/*function to allocate memory for a node*/
NODE getnode(void)
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
    printf("\nOut of memory");
    exit(0);
    }
    return x;
}


/*function to reverse the nodes in a list*/
NODE reverse(NODE start)
{
    NODE cur,temp;
    cur=NULL;
    while(start!=NULL)
    {
    temp=start;
    start=start->link;
    temp->link=cur;
    cur=temp;
    }
    return cur;
}
```

/*function to concatenate two list*/

```
NODE concatenate(NODE first, NODE sec)
{
    NODE  cur;
    if(first==NULL)
    return sec;
    if(sec==NULL)
    return first;
    cur=first;
    while(cur->link!=NULL)
    {
    cur=cur->link;
   }
    cur->link=sec;
    return first;
}


NODE insert_rear(int item,NODE first)
{
    NODE  temp;
    NODE cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
    return temp;
    cur=first;
    while(cur->link!=NULL)
    {
    cur=cur->link;
    }
    cur->link=temp;
    return first;
}
```

/*function to display nodes in the list*/

```
void display(NODE first)
{
        NODE temp;
        if(first==NULL)
        {
        printf("\nList is empty");
        return;
        }
        //printf("\nThe contents of the list:");
        temp=first;
        while(temp!=NULL)
        {
        printf("%d\t",temp->info);
        temp=temp->link;
        }
        printf("\n");
}


void main()
{
        NODE first=NULL,start=NULL,list1=NULL,list2=NULL;
        int choice,item,n,pos,i;
        //clrscr();

        for(;;)
        {
             printf("\n1.Concatenation\n 2.Reverse a list\n3.Exit\n");
              printf("\nEnter the choice\n");
             scanf("%d",&choice);
             switch(choice)
             {
              case 1: printf("\nEnter the no of nodes in the first list\n");
                          scanf("%d",&n);
                         for(i=0;i<n;i++)
                         {
                                     printf("enter the item\n" );
                                     scanf("%d",&item);
                                     list1=insert_rear(item,list1);
                                                }
                                     printf("enter the no of nodes in the second list\n");
                                     scanf("%d",&n);
                                     for(i=0;i<n;i++)
                                     {
                                              printf("enter the item\n");
                                              scanf("%d",&item);
```

```
                                    list2=insert_rear(item,list2);
                            }
                            first= concatenate(list1,list2);
                            printf("The concatenated list is\n");
                            display(first);
                            break;

            case 2: printf("Enter the no of nodes in the list\n");
                    scanf("%d",&n);
                    for(i=0;i<n;i++)
                    {
                            printf("enter the item\n" );
                            scanf("%d",&item);
                            start=insert_rear(item,start);
                    }
                    printf("\nThe given list is:");
                    display(start);

                    start=reverse(start);

                    printf("\nThe reversed list is\n");
                    display(start);
                    break;

          default: exit(0);
          }
      }
}
```

***********************************OUTPUT*********************************

```
1.Concatenation
 2.Reverse a list
3.Exit

Enter the choice
1

Enter the no of nodes in the first list
2
enter the item
10
enter the item
20
```

```
enter the no of nodes in the second list
3
enter the item
100
enter the item
200
enter the item
300
The concatenated list is
10      20      100     200     300

1.Concatenation
 2.Reverse a list
3.Exit

Enter the choice
2
Enter the no of nodes in the list
4
enter the item
12
enter the item
13
enter the item
14
enter the item
15

The given list is:12    13      14      15

The reversed list is
15      14      13      12

1.Concatenation
 2.Reverse a list
3.Exit

Enter the choice
```

Program 12:

Write a c program to support the following operations on a Doubly linked list where each node consists of integers:
1. Create a doubly linked list by adding each node at the front.
2. Insert a new node to the right of the node whose key value read as input.
3. To delete all the nodes whose info is same as key item.
4. Display the content of the list.

```
#include<stdio.h>
//#include<alloc.h>
//#include<conio.h>
#include<stdlib.h>

/*structure declaration for a node of DLL*/
struct node
{
	struct node *llink;
	int info;
	struct node *rlink;
};
typedef struct node *NODE;

/*function to allocate memory for a node*/
NODE getnode(void)
{
	NODE x;
	x=(NODE)malloc(sizeof(struct node));
	if(x==NULL)
	{
		printf("Out of memory..\n");
		exit(0);
	}
	return x;
}

/*function to display the node information in a DLL*/
void display(NODE head)
{
	NODE temp;
	if(head->rlink==head)
	{
		printf("DList is Empty...\n");
		return;
```

```
        }
        printf("The contents of the DList are:\n");
        for(temp=head->rlink;temp!=head;temp=temp->rlink)
        {
                printf("%d ",temp->info);
        }
        printf("\n");
}
```

/*function to insert a node at the front end of DLL*/

```
NODE insert_front(int item,NODE head)
{
        NODE temp,cur;
        temp=getnode();
        temp->info=item;
        cur=head->rlink;
        head->rlink=temp;
        temp->llink=head;
        temp->rlink=cur;
        cur->llink=temp;
        return head;
}
```

/*function to insert a node to the right of the specified key element in the DLL*/

```
NODE insert_right(int item,NODE head)
{
        NODE prev,cur,next,temp;
        if(head->rlink==head)
        {
                printf("List Empty...\n");
                return head;
        }
        cur=head->rlink;
        while(cur!=head && item!=cur->info)
                cur=cur->rlink;
        if(cur==head)
        {
                printf("Key not found...\n");
                return head;
        }
        next=cur->rlink;
        printf("Enter the item to insert towards right of the %d\n",item);
        temp=getnode();
        scanf("%d",&temp->info);
```

```
	cur->rlink=temp;
	temp->llink=cur;
	next->llink=temp;
	temp->rlink=next;
	return head;
}
```

/*function to delete all the nodes whose info is specified from DLL*/

```
NODE delete_all(int item,NODE head)
{
	NODE prev,cur,next;
	int count;
	if(head->rlink==head)
	{
		printf("List Empty...\n");
		return head;
	}
	count=0;
	cur=head->rlink;
	while(cur!=head)
	{
		if(item!=cur->info)
			cur=cur->rlink;
		else
		{
			count++;
			prev=cur->llink;
			next=cur->rlink;
			prev->rlink=next;
			next->llink=prev;
			free(cur);
			cur=next;
		}
	}
	if(count==0)
		printf("Key not found...\n");
	else
		printf("key found at %d positions and are deleted..\n",count);
	return head;
}

void main()
{
	NODE head;
```

```
        int choice,item;
        //clrscr();
        head=getnode();
        head->rlink=head;
        for(;;)
        {
                printf("1:Inser_front\n2:Insert right of key\n");
                printf("3:Delete same key\n4:Display\n5:Exit\n");
                printf("Enter the choice\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("Enter the item to be inserted:");
                                scanf("%d",&item);
                                head=insert_front(item,head);
                                break;
                        case 2: printf("\nEnter the key:");
                                scanf("%d",&item);
                                head=insert_right(item,head);
                                break;
                        case 3: printf("\nEnter the key to be deleted:");
                                scanf("%d",&item);
                                head=delete_all(item,head);
                                break;
                        case 4: display(head);
                                break;

                        default:
                                printf("Invalid choice...\n");
                                exit(0);
                }
        }
}
```

```
**********************************OUTPUT**********************************
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
1
Enter the item to be inserted:10
1: Inser_front
```

```
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
1
Enter the item to be inserted:20
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
1
Enter the item to be inserted:30
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
4
The contents of the DList are:
30 20 10
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
2

Enter the key:30
Enter the item to insert towards right of the 30
40
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
4
The contents of the DList are:
30 40 20 10
```

```
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
1
Enter the item to be inserted:20
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
4
The contents of the DList are:
20 30 40 20 10
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
3

Enter the key to be deleted:20
key found at 2 positions and are deleted..
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
4
The contents of the DList are:
30 40 10
1: Inser_front
2: Insert right of key
3: Delete same key
4: Display
5: Exit
Enter the choice
5
Invalid choice...
```

Program 13:

Write a C program
i) To create a Binary tree based on given direction.
ii) To search for an item
iii) To get the exact copy of a tree   iv) To display the elements based on its level from L-R

```
#include<stdio.h>
//#include<alloc.h>
#include<conio.h>
//#include<process.h>
#include<string.h>
int flag=0;

/*structure declaration for a node*/
struct node
{
        int info;
        struct node *llink,*rlink;
};
typedef struct node *NODE;

/*function to allocate memory for a node*/
NODE getnode()
{
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
         if(x==NULL)
         {
                 printf("out ofmemory\n");
                exit(0);
         }
         return x;
}

/*function to create a Binary tree based on given direction*/
NODE insert(int item,NODE root)
{
         NODE temp,cur,prev;
        char direction[10];
        int i;
        temp=getnode();
        temp->info=item;
        temp->llink=temp->rlink=NULL;
         if(root==NULL)
                return temp;
         printf("give the direction you want to insert \n");
         scanf("%s",direction);

         prev=NULL;
         cur=root;

        for(i=0;i<strlen(direction) && cur!=NULL;i++)
        {
                prev=cur;
                if(direction[i]=='l')
                        cur=cur->llink;
                else
                        cur=cur->rlink;
        }
I       f(cur!=NULL || i!=strlen(direction))
        {
                printf("insertion is not possible\n");
                free(temp);
                return root;
         }
        if(direction[i-1]=='l')
                 prev->llink=temp;
         else
                 prev->rlink=temp;
 return root;
}
```

/*function to display the elements based on its level from L-R*/

```
void display_level(NODE root)
{
        NODE q[20],cur;
        int front=0,rear=-1;
        q[++rear]=root;
        while(front<=rear)
        {
                 cur=q[front++];
                 printf("%d\n",cur->info);
                 if(cur->llink!=NULL)
                        q[++rear]=cur->llink;
                if(cur->rlink!=NULL)
                         q[++rear]=cur->rlink;
        }
        printf("\n");
}
```

/*function To get the exact copy of a tree*/

```
NODE copy_tree(NODE root)
{
         NODE temp;
        if(root==NULL)
                return NULL;
        temp=getnode();
        temp->info=root->info;
        temp->llink=copy_tree(root->llink);
         temp->rlink=copy_tree(root->rlink);
         return temp;
}
```

/*function to search for a node in a tree*/

```
void search(int item,NODE root)
{
        if(root!=NULL)
        {
                search(item,root->llink);
                if(item==root->info)
                {
                        flag=1;
                        return;
                }
                search(item,root->rlink);
        }
}

void main()
{
        NODE copied,root=NULL;
        int ch,item;
        //clrscr();
        do
        {
                printf("1.insert\n 2.search\n 3.exact copy\n 4.display\n 5.exit\n");
                printf("enter the choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("enter the item\n");
                                 scanf("%d",&item);
                                root=insert(item,root);
                                break;
                        case 2:if(root==NULL)
                                        printf("tree is empty\n");
                                else
                                 {
                                        printf("enter the key\n");
                                        scanf("%d",&item);
                                        flag=0;
                                        search(item,root);
                                        if(flag==0)
                                                printf("key not found\n");
                                         else
                                                 printf("key found\n");
                                }
                                break;

                        case 3:if(root==NULL)
                                        printf("tree is empty\n");
                                else
                                {
                                        copied=copy_tree(root);
                                         printf("the copied tree is\n");
                                         display_level(copied);
                                }
                                break;

                        case 4:if(root==NULL)
                                        printf("tree is empty\n");
                                 else
                                {
                                        printf("level order traversal of the tree\n");
                                        display_level(root);
                                }
                                break;
                        default: exit(0);
                 }
        } while(ch!=0);
        getch();
}
```

```
*********************************OUTPUT************************************
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
100
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
90
give the direction you want to insert
l
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
80
give the direction you want to insert
r
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
70
give the direction you want to insert
l
insertion is not possible
```

```
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
70
give the direction you want to insert
ll
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
60
give the direction you want to insert
lr
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
50
give the direction you want to insert
ll
insertion is not possible
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
50
give the direction you want to insert
```

```
rl
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
1
enter the item
40
give the direction you want to insert
rr
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
4
level order traversal of the tree
100
90
80
70
60
50
40

1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
2
enter the key
80
key found
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
```

```
2
enter the key
20
key not found
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
3
the copied tree is
100
90
80
70
60
50
40
1.insert
 2.search
 3.exact copy
 4.display
 5.exit
enter the choice
5
```

Program 14:

Write a C program

- **i) To construct a BST of items (No Duplicates)**
- **ii) To search an item in BST**
- iii) **To display the elements of the tree using Inorder, Preorder, & Postorder traversal methods**.

```
#include<stdio.h>
//#include<alloc.h>
#include<conio.h>
#include<stdlib.h>
//#include<process.h>
int flag;

/*structure declaration for a node*/
struct node
{
	int info;
	 struct node *llink,*rlink;
};
typedef struct node *NODE;

/*function to allocate memory for a node*/
NODE getnode()
{
	NODE x;
	x=(NODE)malloc(sizeof(struct node));
	if(x==NULL)
	{
		printf("out of memory\n");
		exit(0);
	}
	return x;
}

/*function to insert a node into BST*/
NODE insert(NODE root,int item)
{
	NODE temp,cur,prev;
	temp=getnode();
	temp->info=item;
	temp->llink=temp->rlink=NULL;
	if(root==NULL)
```

```
                return temp;

        prev =NULL;
        cur=root;

        while(cur!=NULL)
        {
                prev=cur;
                 if(item==cur->info)
                {
                        printf("duplicate items are not allowed\n");
                        free(temp);
                        return root;
                 }
                 if(item<cur->info)
                        cur=cur->llink;
                 else
                         cur=cur->rlink;
         }
        if(item<prev->info)
                 prev->llink=temp;
        else
                 prev->rlink=temp;
        return root;
}
```

/*function to search for a node in a tree*/

```
void search(NODE root,int item)
{
        if(root!=NULL)
        {
                if(item==root->info)
                {
                        flag=1;
                        return;
                }
                if(item<root->info)
                        search(root->llink,item);
                 else
                         search(root->rlink,item);
        }
 }
```

```
/*function for inorder traversal*/
 void in_order(NODE root)
 {
            if(root!=NULL)
            {
                        in_order(root->llink);
                        printf("%d\t",root->info);
                         in_order(root->rlink);
            }
 }

/*function for preorder traversal*/
 void pre_order(NODE root)
 {
            if(root!=NULL)
            {
                         printf("%d\t",root->info);
                         pre_order(root->llink);
                         pre_order(root->rlink);
             }
 }


 /*function for postorder traversal*/
void post_order(NODE root)
 {
            if(root!=NULL)
            {
                        post_order(root->llink);
                        post_order(root->rlink);
                        printf("%d\t",root->info);
            }
 }



 void main()
 {
            NODE root=NULL;
            int ch,item;
            //clrscr();
            for(;;)
            {
```

```
                     printf("\n1.BST Insertion\n 2.search\n 3.preorder\n 4.inorder\n 5.postorder\n
                     6.exit\n");
                    printf("\nEnter your choice:");
                     scanf("%d",&ch);
                    switch(ch)
                    {
                            case 1:printf("\nEnter the item to be inserted:");
                                    scanf("%d",&item);
                                    root=insert(root,item);
                                    break;

                            case 2:printf("\nEnter the item to be searched:");
                                    scanf("%d",&item);
                                    flag=0;
                                    search(root,item);
                                    if(flag==0)
                                            printf("unsuccessfull search\n");
                                    else
                                            printf("successfull search\n");
                                    break;

                            case 3:printf("pre traversal is\n");
                                    pre_order(root);
                                    break;
                            case 4:printf("in traversal is\n");
                                    in_order(root);
                                     break;

                            case 5:printf("post traversal is\n"  );
                                    post_order(root);
                                    break;

                            default:exit(0);
                     }
            }

    }
```

```
*********************************OUTPUT*********************************

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:1

Enter the item to be inserted:100

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:1

Enter the item to be inserted:120

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:1

Enter the item to be inserted:90

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:1
Enter the item to be inserted:95

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:1

Enter the item to be inserted:70

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:1

Enter the item to be inserted:110

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:1

Enter the item to be inserted:125

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:3
pre traversal is
```

```
100     90      70      95      120     110     125
1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:4
in traversal is
70      90      95      100     110     120     125
1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:5
post traversal is
70      95      90      110     125     120     100
1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:2

Enter the item to be searched:110           20
successfull search

1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:2

Enter the item to be searched:130
unsuccessfull search
```

```
1.BST Insertion
 2.search
 3.preorder
 4.inorder
 5.postorder
 6.exit

Enter your choice:6
```