NAME : YOGESH T

USN : 1BM19CS188

SUBJECT : DATA STRUCTURES

ACADEMIC YEAR : 2020 - 2021

Program : 1

```
#include<stdio.h>

#include<stdlib.h>

#define SIZE 10


void push(int );

void pop();

void display();


int value,choice,stack[SIZE],top= -1;


int main()

{

   printf("(1) - push\n");

   printf("(2) - pop\n");

   printf("(3) - display\n\n");

   printf("enter the choice\n");

   scanf("%d",&choice);
```

```
    while(choice!=4)
    {
        switch(choice)
        {
        case 1:
            printf("enter the value to be inserted\n");
            scanf("%d",&value);
            push(value);
            break;

        case 2:
            pop();
            break;

        case 3:
            display();
            break;
        }
    printf("\n");
    printf("enter choice 4 to exit & other to continue\n");
    scanf("%d",&choice);
    }
    return 0;
}
void push(int value)
```

```
{
   if (top==SIZE -1)
      printf("Stack overflow\n");
   else
   {
      top+=1;
      stack[top]=value;
      printf("insertion successful\n");
   }
}

void pop()
{
   if(top==-1)
      printf("Stack Underflow\n");
   else
   {
      printf("Value %d is deleted successfully",stack[top]);
      top--;
   }
}

void display()
{
   if (top==-1)
```

```
        printf("stack is empty\n");
    else
    {
        printf("elements are:\n");

        for(int i=top;i>=0;i--)
            printf("%d\n",stack[i]);
    }
}
```

OUTPUT :-

```
Output :

(1) - push
(2) - pop
(3) - display

enter the choice
1
enter the value to be inserted
2
insertion successful

enter choice 4 to exit & other to continue
1
enter the value to be inserted
3
insertion successful

enter choice 4 to exit & other to continue
3
elements are:
3
2

enter choice 4 to exit & other to continue
2
Value 3 is deleted successfully
enter choice 4 to exit & other to continue
4

Process returned 0 (0x0)   execution time : 32.071 s
Press any key to continue.
```

Program : 2

```
#include<stdio.h>

#include<ctype.h>
```

```
#define SIZE 50

char stack[SIZE];

int top=-1;

push(char elem)

{

stack[++top]=elem;

}

char pop()

{

 return(stack[top--]);

 }

 int pr(char symbol)

 {

 if(symbol== '^')

 {

 return(3);

 }

 else if(symbol== '*' || symbol== '/' )

 {

 return(2);

 }

 else if(symbol== '+' || symbol== '-')

 {

 return(1);

 }
```

```
else {
return(0);
}
}
void main()
{
char infix[50],postfix[50],ch,elem;
int i=0,k=0;
printf("enter Infix expression");
scanf("%s",infix);
push('#');
while((ch=infix[i++])!='\0' )
{
if(ch == '(') push(ch);
else
if(isalnum(ch)) postfix[k++]=ch;
else
if(ch==')')
{ while(stack[top] !='(')
postfix[k++]=pop();
elem=pop();
}
else
{
while(pr(stack[top]) >=pr(ch))
```

```
postfix[k++]=pop();
push(ch);
}
}
while(stack[top]!='#')
 postfix[k++]=pop();
 postfix[k]='\0';
 printf("\nPostfixexpression=%s\n",postfix);
 }
```

OUTPUT :-

```
OUTPUT :

enter Infix expression
((A+B)*C-D)

Postfixexpression = AB+C*D-
```

Program : 3

```
#include <stdio.h>
#define MAX 20  // Changing this value will change length of array


int queue[MAX];
int front = -1, rear = -1;
void insert(void);
int delete_element(void);
```

```
int peek(void);
void display(void);



int main()
 {
   int option, val;
   do
   {
       printf("***** MAIN MENU *****\n");
       printf("1. Insert an element\n");
       printf("2. Delete an element\n");
       printf("3. Peek\n");
       printf("4. Display the queue\n");
       printf("5. EXIT\n");
       printf("Enter your option : \n");
       scanf("%d", &option);
      switch(option)
      {
         case 1:
          insert();
          break;
         case 2:
          val = delete_element();
          if (val != -1)
```

```
            printf("The number deleted is : %d\n\n", val);
          break;
         case 3:
          val = peek();
          if (val != -1)
          printf("The first value in queue is : %d\n\n", val);
         break;
         case 4:
          display();
          break;
      }
    }while(option != 5);
    return 0;
}
void insert()
{
  int num;
  printf("Enter the number to be inserted in the queue : \n");
  scanf("%d", &num);
  if(rear == MAX-1)
  printf("OVERFLOW\n");
  else if(front == -1 && rear == -1)
  front = rear = 0;
  else
  rear++;
```

```
  queue[rear] = num;
}
int delete_element()
{
  int val;
  if(front == -1 || front>rear)
  {
     printf("UNDERFLOW\n\n");
     return -1;
  }
  else
  {
     val = queue[front];
     front++;
     if(front > rear)
     front = rear = -1;
     return val;
  }
}
int peek()
{
  if(front==-1 || front>rear)
  {
     printf("QUEUE IS EMPTY\n\n");
     return -1;
```

```
  }
  else
  {
     return queue[front];
  }
}
void display()
{
  int i;
  printf("\n");
  if(front == -1 || front > rear)
     printf("QUEUE IS EMPTY\n\n");
  else
  {
     for(i = front;i <= rear;i++)
     printf("%d\t", queue[i]);
  }
}
```

<u>OUTPUT :-</u>

```
OUTPUT :

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
1
Enter the number to be inserted in the queue :
1
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
1
Enter the number to be inserted in the queue :
2
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
1
Enter the number to be inserted in the queue :
3
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
1
Enter the number to be inserted in the queue :
4
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
4

1       2       3       4       ***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
2
The number deleted is : 1

***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
1
Enter the number to be inserted in the queue :
3
***** MAIN MENU *****
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option :
```

Program : 4

```
#include<stdio.h>
# define max 3
void enqueue( int q[], int *f, int *r)
{
	if(*r-*f==max-1|| *r==*f-1)
	printf(" Queue is full\n\n");
	else
		{
		if(*r==-1)
			*f=*r=0;
		else
			*r=(*r+1)%max;
		printf("Enter the element:\n");
		scanf("%d", (&q[*r]));

		}
}


void dequeue(int q[], int *f, int *r)
{
	if (*f ==-1)
		printf(" Queue is empty\n\n");
	else
```

```
		{
		printf("%d is deleted\n", q[*f]);
		if (*f==*r)
			*f=*r=-1;
		else
			*f=(*f+1)%max;
		}
}

void display (int q[], int *f, int *r)
{
	if(*f==-1)
		printf("Queue is empty\n\n");
	else
		{
			for(int i=*f;;i++)
			{
				i=i%max;
				printf("%d ", q[i]);
				if (*r==i)
					break;
			}
		printf("\n");
		}
}


int main()
```

```
{
        int choice, f=-1, r=-1, q[max];
        do
        {

                printf(" 1: Insert \n 2:Delete \n 3: Display\n 4: Exit\n");
                printf("Enter your choice\n");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1: enqueue(q, &f, &r);
                                break;
                        case 2: dequeue(q, &f, &r);
                                break;
                        case 3: display(q, &f, &r);
                                break;
                        case 4:
                                break;
                        default: printf("INVALID CHOICE\n");
                }
  }while(choice!=4);
}
```

OUTPUT :-

```
OUTPUT :
1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
1
Enter the element:
2
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
1
Enter the element:
3
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
1
Enter the element:
4
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
1
 Queue is full

 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
3
2 3 4
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
2
2 is deleted
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
3
3 4
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
```

Program : 5 &6 in 1

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

typedef struct node{
  int data;
  struct node *next;
}node;

node *head=NULL;

void add_at_end()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));

   printf("Enter the node element\n");
   scanf("%d",&temp->data);
    temp->next=NULL;
   if(head==NULL)
  {
  head=temp;
  }
  else
  {
```

```
    node *ptr=head;
    while(ptr->next!=NULL)
   {
    ptr=ptr->next;
   }
   ptr->next=temp;
  }
}

void add_at_begin()
{
 node *temp;
temp=(node *)malloc(sizeof(node));
 printf("Enter node element\n");
 scanf("%d",&temp->data);
 temp->next=NULL;

 if(head==NULL)
 {
     head=temp;
 }
 else
 {
  temp->next=head;
  head=temp;
 }
}
```

```
int length()
{
  node *ptr;
  ptr=head;
  int i=0;

  while(ptr!=NULL)
  {
   i++;
   ptr=ptr->next;
  }
return i;
}

void add_after(){

node *ptr,*temp;
int loc,i=1;
printf("Enter the location");
scanf("%d",&loc);

if(loc>length())
{
  printf("Invalid location. The list has %d nodes",length());
}
else
{
```

```
ptr=head;
while(i<loc)
{
 ptr=ptr->next;
 i++;
}

temp=(node *)malloc(sizeof(node));
printf("Enter the node element\n");
scanf("%d",&temp->data);
temp->next=NULL;

temp->next=ptr->next;
ptr->next=temp;
}
}

void delete()
{
int loc;
node *temp;
printf("Enter the locatin of node to be deleted\n");
scanf("%d",&loc);

if (loc>length())
{
  printf("There is no such node\n");
}
```

```
else if (loc==1)
{
 temp=head;
head=temp->next;
temp->next=NULL;
free(temp);
}
else
{
node *ptr=head,*q;
int i=1;
while(i<loc-1)
{
  ptr=ptr->next;
  i++;
}
q=ptr->next;
ptr->next=q->next;
q->next=NULL;
free(q);
}
}


void display()
{
 node *temp=head;
 if(temp==NULL)
```

```
 {
   printf("No nodes in the list\n");
 }
 else
 {
   while(temp!=NULL)
   {
     printf("%d\n",temp->data);
     temp=temp->next;
   }
 }
}




int main()
{

 int op,len;
 while(1)
{ printf("Enter the operation\n1.Add in begin\n2.Add at end\n");
  printf("3.Add after a node\n4.Delete node\n5.Display\n6.Length of list\n7.Exit\n");
  scanf("%d",&op);
  switch (op)
  {
  case 1:add_at_begin();
    break;
  case 2: add_at_end();
```

```
  break;
 case 3: add_after();
  break;
 case 4: delete();
  break;
 case 5: display();
  break;
 case 6: len=length();
        printf("The length is %d\n",len);
  break;
 case 7: exit(0);
   break;
 default: printf("No such operation\n");
 }
}
return 0;
}
```

<u>OUTPUT :-</u>

```
OUTPUT:

Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
1
Enter node element
1
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
1
Enter node element
2
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
1
Enter node element
3
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
1
Enter node element
4
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
1
Enter node element
5
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
1
```

```
1
Enter node element
6
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
5
6
5
4
3
2
1
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
4
Enter the locatin of node to be deleted
3
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
5
6
5
3
2
1
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
1
Enter node element
2
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
5
2
6
5
3
2
1
Enter the operation
```

```
5
2
6
5
3
2
1
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
3
Enter the location1
Enter the node element
2
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
5
2
2
6
5
3
2
1
Enter the operation
1.Add in begin
2.Add at end
3.Add after a node
4.Delete node
5.Display
6.Length of list
7.Exit
6
The length is 7
```

Program : 7 & 8 in 1

```
#include<stdio.h>

#include<stdlib.h>


struct node
```

```
{
	int data;
	struct node *next;
};
struct node *head;
struct node *head2;

//stack operations
void push()
{
	struct node *ptr;
	int new_data;
	ptr = (struct node *)malloc(sizeof(struct node));

	if(ptr == NULL)
	{
		printf("\nOVERFLOW!!!");
	}
	else
	{
		printf("\nEnter the Value to be inserted:");
		scanf("%d",&new_data);
		ptr->data = new_data;
		ptr->next = head;
		head = ptr;
		printf("\nNODE INSERTED AT THE TOP OF THE STACK\n");
	}
}
```

```
void pop()
{
	struct node *ptr;
	if(head == NULL)
	{
		printf("EMPTY LIST!!!");
	}
	else
	{
		ptr = head;
		head = ptr->next;
		free(ptr);
		printf("\nNODE DELETED FROM TOP OF THE STACK\n");
	}
}

//queue operations
void enqueue()
{
	struct node *ptr,*temp;
	int new_data;
	ptr = (struct node *)malloc(sizeof(struct node));

	printf("\nEnter the Value to be inserted:");
	scanf("%d",&new_data);
	ptr->data = new_data;
	if(head == NULL)
	{
```

```
		ptr->next = NULL;
		head = ptr;
		printf("\nNODE INSERTED AT REAR OF THE QUEUE\n");
	}
	else
	{
		temp = head;
		while(temp->next != NULL)
		{
			temp = temp->next;
		}
		temp->next = ptr;
		ptr->next = NULL;
		printf("\nNODE INSERTED AT REAR OF THE QUEUE\n");
	}
}
void dequeue()
{
	struct node *ptr;
	if(head == NULL)
	{
		printf("EMPTY LIST!!!");
	}
	else
	{
		ptr = head;
		head = ptr->next;
		free(ptr);
```

```
		printf("\nNODE DELETED FROM FRONT OF THE QUEUE\n");
	}
}

//Display List
void display()
{
	struct node *ptr;
	ptr = head;
	if(ptr == NULL)
	{
		printf("EMPTY LIST!!!INSERT FEW ELEMENTS!!");
	}
	else
	{
		printf("\n\nLIST-->");
		while(ptr != NULL)
		{
			printf("\t%d",ptr->data);
			ptr = ptr->next;
		}
	}
}

//sort Linked list in ascending order
void sort()
{
	struct node *ptr = head;
```

```
        struct node *temp = NULL;
        int i;

        if(head == NULL)
        {
                return;
        }
        else
        {
                while(ptr != NULL)
                {
                        temp = ptr->next;
                        while(temp != NULL)
                        {
                                if(ptr->data >temp->data)
                                {
                                        i = ptr->data;
                                        ptr->data = temp->data;
                                        temp->data = i;
                                }
                                temp = temp->next;
                        }
                        ptr = ptr->next;
                }
        }
}

//reverse Linked List
```

```
void reverse()
{
        struct node *prev = NULL;
        struct node *next = NULL;
        struct node *ptr = head;
        while(ptr != NULL)
        {
                next = ptr->next;
                ptr->next = prev;
                prev = ptr;
                ptr = next;
        }
        head = prev;
}

//create list
struct node *create_list(struct node *head)
{
        struct node *ptr,*temp;
        int i,n,new_data;

   printf("\nEnter the number of nodes : ");
   scanf("%d",&n);
        head = NULL;
        if(n == 0)
        {
                return head;
        }
```

```
	for(i=1;i<=n;i++)
	{
		ptr = (struct node *)malloc(sizeof(struct node));
		printf("Enter the element to be inserted : ");
		scanf("%d",&new_data);
		ptr->data = new_data;
		if(head == NULL)
		{
			ptr->next = NULL;
			head = ptr;
		}
		else
		{
			temp = head;
			while(temp->next != NULL)
			{
				temp = temp->next;
			}
			temp->next = ptr;
			ptr->next = NULL;
		}
	}
	return head;
}



//concatenate two lists
struct node *concatenate(struct node *head, struct node *head2)
```

```
{
	struct node *ptr;
	if(head == NULL)
	{
		head = head2;
		return head;
	}
	if(head2 == NULL)
	{
		return head;
	}
	ptr = head;
	while(ptr->next != NULL)
	{
		ptr = ptr->next;
	}
	ptr->next = head2;
	return head;
}




int main()
{
	int choice = 0;
	while(1)
	{
		printf("\n\n*****************MENU***************\n");
```

```
printf("Choose an option from the list:");
printf("\n-----STACK OPERATIONS-----\n[1]PUSH\n[2]POP");
printf("\n-----QUEUE OPERATIONS-----\n[3]ENQUEUE\n[4]DEQUEUE");
printf("\n--------------------------");
printf("\n[5]DISPLAY\n[6]SORT\n[7]REVERSE\n[8]CONCATENATION\n[9]EXIT\n");
printf("\nEnter your choice:");
scanf("%d",&choice);
switch(choice)
{
        case 1: push();
                        break;
        case 2: pop();
                        break;
        case 3: enqueue();
                        break;
        case 4: dequeue();
                        break;
        case 5: display();
                        break;
        case 6: sort();
                        printf("\nSorted List::");
                        display();
                        break;
        case 7: reverse();
                        printf("\nReversed List::");
                        display();
                        break;
        case 8: printf("\nCreate a Second list-->");
```

```
                    head2 = create_list(head2);
                    printf("\nList1:");
                    display();
                    struct node *ptr;
                    ptr = head2;
                    if(ptr == NULL)
                    {
                        printf("LIST2 IS EMPTY!!!");
                    }
                    else
                    {
                        printf("\n\nLIST2-->");
                        while(ptr != NULL)
                        {
                            printf("\t%d",ptr->data);
                            ptr = ptr->next;
                        }
                    }
                    head = concatenate(head,head2);
                    printf("\n\nConcatenated List:");
                    display();
                    break;
            case 9:  exit(1);
            default:
                printf("\nINVALID CHOICE!!!\n");

        }
}
```

}

OUTOUT :

```
*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:1

Enter the Value to be inserted:1

NODE INSERTED AT THE TOP OF THE STACK


*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:1

Enter the Value to be inserted:2

NODE INSERTED AT THE TOP OF THE STACK


*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:1

Enter the Value to be inserted:4

NODE INSERTED AT THE TOP OF THE STACK


*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:1

```

```
Enter your choice:1

Enter the Value to be inserted:3

NODE INSERTED AT THE TOP OF THE STACK


****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:5


LIST--> 3       4       2       1

****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:6

Sorted List::

LIST--> 1       2       3       4

****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:7

Reversed List::

LIST--> 4       3       2       1

****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT
```

```
*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:8

Create a Second list-->
Enter the number of nodes : 2
Enter the element to be inserted : 5
Enter the element to be inserted : 6

List1:

LIST--> 4       3       2       1

LIST2-->        5       6

Concatenated List:

LIST--> 4       3       2       1       5       6

*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:6

Sorted List::

LIST--> 1       2       3       4       5       6

*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:2

NODE DELETED FROM TOP OF THE STACK


*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
```

```
LIST--> 2       3       4       5       6

*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:3

Enter the Value to be inserted:1

NODE INSERTED AT REAR OF THE QUEUE


*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:5


LIST--> 2       3       4       5       6       1

*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:
6

Sorted List::

LIST--> 1       2       3       4       5       6

*****************MENU****************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
--------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:
```

Program : 9

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

struct node
{
	struct node *prev;
	int data;
	struct node *next;
};
struct node *head;
struct node *last;

//Create DLL
void create_list()
{
	struct node *ptr;
	int i,n,new_data;
	printf("\nEnter the number of nodes:");
	scanf("%d",&n);

	if(n>=1)
	{
		head = (struct node *)malloc(sizeof(struct node));
		if(head != NULL)
```

```
{
        printf("\nEnter the value to be inserted for Node 1 :\t");
        scanf("%d",&new_data);

        head->data = new_data;
        head->prev = NULL;
        head->next = NULL;

        last = head;

        for(i=2;i<=n;i++)
        {
                ptr = (struct node *)malloc(sizeof(struct node));

                if(ptr != NULL)
                {
                        printf("Enter the value to be inserted for Node %d :\t",i);
                        scanf("%d",&new_data);

                        ptr->data = new_data;
                        ptr->prev = last;
                        ptr->next = NULL;

                        last->next = ptr;
                        last = ptr;
                }
        }
        printf("\n\nLinked List Created!!");
```

```
		}
	}
	else
	{
		printf("\n\nInvalid!!Enter valid number of nodes!!");
	}
}

//display
void display_list()
{
	struct node *ptr = head;
	if(ptr == NULL)
	{
		printf("\nList is Empty!!");
	}
	else
	{
		printf("\n\nLIST-->");
		while(ptr != NULL)
		{
			printf("\t%d",ptr->data);
			ptr = ptr->next;
		}
	}
}

//Insert at the left of a given node
```

```
void insert_left()
{
	int i,pos,new_data;
	struct node *ptr,*temp;
	ptr = (struct node *)malloc(sizeof(struct node));

	printf("\nEnter the Node to insert the value: ");
	scanf("%d",&pos);
	printf("\nEnter the value to be inserted: ");
	scanf("%d",&new_data);

	ptr->data = new_data;

	if(head == NULL)
	{
		printf("\nList is empty!!");
	}
	else
	{
		temp = head;
		i=1;
		while(i<pos-1 && temp!=NULL)
		{
			temp = temp->next;
			i++;
		}
		if(pos == 1)
		{
```

```
        ptr->next = head;
        ptr->prev = NULL;
        head->prev = ptr;
        head = ptr;
        printf("\n\nNode Inserted at %d position!!",pos);
}
else if(temp == last)
{
        ptr->next = NULL;
        ptr->prev = last;
        last->next = ptr;
        last = ptr;
        printf("\n\nNode Inserted at %d position!!",pos);
}
else if(temp != NULL)
{
        ptr->next = temp->next;
        ptr->prev = temp;
        if(temp->next != NULL)
        {
                temp->next->prev = ptr;
        }
        temp->next = ptr;
        printf("\n\nNode Inserted at %d position!!",pos);
}
else
{
        printf("\n\nInvalid Position!!");
```

```
                        }
            }
}

//Delete node by Value
void delete()
{
  struct node* temp = head;
  struct node* ptr = (struct node*) malloc(sizeof(struct node));
  int val;
  printf("\nEnter the Value to be deleted: ");
  scanf("%d",&val);

  if(temp->next == NULL)
  {
    head = NULL;
    free(temp);
    printf("\n\nValue %d, deleted \n",val);
    return;
  }

  if(temp!=NULL && temp->data == val)
  {
    head = temp->next;
    head->prev = NULL;
    free(temp);
    printf("\n\nValue %d, deleted ",val);
            return;
```

```
  }
  while(temp!=NULL && temp->data != val)
  {
    ptr = temp;
    temp = temp->next;
  }
  if(temp==NULL)
  {
    printf("\n\nValue not found");
    return;
  }
  ptr->next = temp->next;

  if(temp->next == NULL)
  {
    printf("\n\nValue %d, deleted \n",val);
    free(temp);
    return;
  }

  struct node* temp2 = (struct node*) malloc(sizeof(struct node));
  temp2 = temp->next;
  temp2->prev = ptr;
  free(temp);
  printf("Value %d, deleted \n",val);
}

int main()
```

```
{
	int choice = 0;
	while(1)
	{
		printf("\n\n*****************MENU***************\n");
		printf("Choose an option from the list:");
		printf("\n[1]CREATE A LIST\n[2]INSERT TO THE LEFT OF A NODE\n[3]DELETE NODE
\n[4]DISPLAY\n[5]EXIT\n");
		printf("\nEnter your choice:");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1: create_list();
				break;
			case 2: insert_left();
				break;
			case 3: delete();
				break;
			case 4: display_list();
				break;
			case 5: exit(1);
			default:
				printf("\nINVALID CHOICE!!!\n");

		}
	}
}
```

OUTPUT :

```

Sir output was a bit long, which required multiple screenshots, so i have copied and pasted the output of output contents


*****************MENU****************
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT

Enter your choice:1

Enter the number of nodes:4

Enter the value to be inserted for Node 1 :     1
Enter the value to be inserted for Node 2 :     2
Enter the value to be inserted for Node 3 :     3
Enter the value to be inserted for Node 4 :     4


Linked List Created!!

*****************MENU****************
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT

Enter your choice:3

Enter the Value to be deleted: 2
Value 2, deleted


*****************MENU****************
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT

Enter your choice:2

Enter the Node to insert the value: 1

Enter the value to be inserted: 2


Node Inserted at 1 position!!

*****************MENU****************
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT

Enter your choice:4


LIST--> 2       1       3       4

*****************MENU****************
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT

Enter your choice:
```

PROGRAM : 10

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

struct node
{
	int key;
	struct node *left;
	struct node *right;
};
struct node *root;

struct node *create(int data)
{
	struct node *temp;
	temp = (struct node*)malloc(sizeof(struct node));
	temp->key = data;
	temp->left = temp->right = NULL;
	return temp;
}

void insert(struct node *root,struct node *temp)
{
	if(temp->key < root->key)
```

```
        {
                if(root->left != NULL)
                {
                        insert(root->left,temp);
                }
                else
                {
                        root->left = temp;
                }
        }
        if(temp->key > root->key)
        {
                if(root->right != NULL)
                {
                        insert(root->right,temp);
                }
                else
                {
                        root->right = temp;
                }
        }
}

void display(struct node *root)
{
        if(root != NULL)
        {
                display(root->left);
```

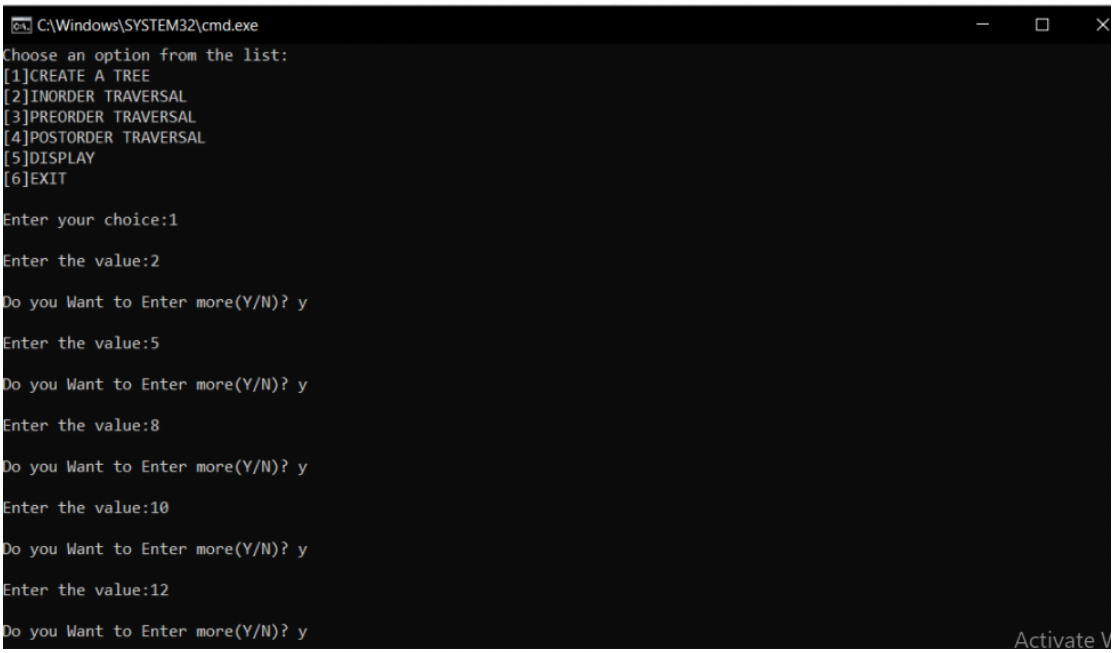
```
		printf("%d\t",root->key);
		display(root->right);
	}
}

void inorder(struct node *root)
{
	if(root != NULL)
	{
		inorder(root->left);
		printf("%d\t",root->key);
		inorder(root->right);
	}
}
void preorder(struct node *root)
{
	if(root != NULL)
	{
		printf("%d\t",root->key);
		preorder(root->left);
		preorder(root->right);
	}
}
void postorder(struct node *root)
{
	if(root != NULL)
	{
		postorder(root->left);
```

```
                postorder(root->right);
                printf("%d\t",root->key);
        }
}
int main()
{
        char ch;
        struct node *temp;
        root = NULL;
        int choice = 0;
        int data;

        while(1)
        {
                printf("\n\n*****************MENU****************\n");
                printf("Choose an option from the list:");
                printf("\n[1]CREATE A TREE\n[2]INORDER TRAVERSAL\n[3]PREORDER
TRAVERSAL\n[4]POSTORDER TRAVERSAL\n[5]DISPLAY\n[6]EXIT\n");
                printf("\nEnter your choice:");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: do{
                                        printf("\nEnter the value:");
                                        scanf("%d",&data);
                                        temp = create(data);
                                        if(root == NULL)
                                        {
```

```
                    root = temp;
                }
                else
                {
                    insert(root,temp);
                }
                printf("\nDo you Want to Enter more(Y/N)? ");
                getchar();
                scanf("%c",&ch);
            }while(ch=='y'||ch=='Y');
            break;
case 2: printf("\nINORDER TRAVERSAL-->\t");
            inorder(root);
            break;

case 3: printf("\nPREORDER TRAVERSAL-->\t");
            preorder(root);
            break;

case 4: printf("\nPOSTORDER TRAVERSAL-->\t");
            postorder(root);
            break;

case 5: display(root);
            break;

case 6: exit(1);
default:
```

```
                    printf("\nINVALID CHOICE!!!\n");

                }

        }

        return 0;

}
```

OUTPUT :-



```
C:\Windows\SYSTEM32\cmd.exe

Enter the value:4

Do you Want to Enter more(Y/N)? n


*****************MENU****************
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT

Enter your choice:5
2       3       4       5       8       10      12

*****************MENU****************
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT

Enter your choice:4

POSTORDER TRAVERSAL-->  4       3       12      10      8       5       2
```

```
C:\Windows\SYSTEM32\cmd.exe

*****************MENU****************
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT

Enter your choice:3

PREORDER TRAVERSAL-->   2       5       3       4       8       10      12

*****************MENU****************
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT

Enter your choice:2

INORDER TRAVERSAL-->    2       3       4       5       8       10      12

*****************MENU****************
Choose an option from the list:
[1]CREATE A TREE
```