

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 :-

```

83  Output :
84
85  (1) - push
86  (2) - pop
87  (3) - display
88
89  enter the choice
90  1
91  enter the value to be inserted
92  2
93  insertion successful
94
95  enter choice 4 to exit & other to continue
96  1
97  enter the value to be inserted
98  3
99  insertion successful
100
101 enter choice 4 to exit & other to continue
102 3
103 elements are:
104 3
105 2
106
107 enter choice 4 to exit & other to continue
108 2
109 Value 3 is deleted successfully
110 enter choice 4 to exit & other to continue
111 4
112
113 Process returned 0 (0x0)   execution time : 32.071 s
114 Press any key to continue.

```

Activate Windows
Go to Settings to activate Windows.

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))

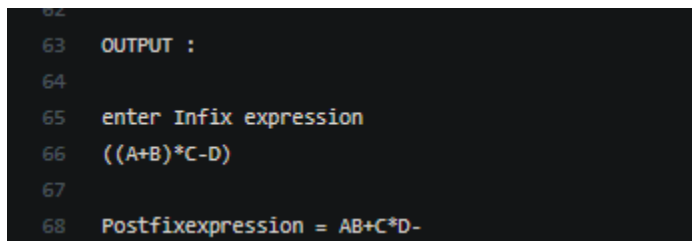
```

```

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

```

OUTPUT :-



```

62
63  OUTPUT :
64
65  enter Infix expression
66  ((A+B)*C-D)
67
68  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]);  
    }  
}
```

OUTPUT :-

```

101
102 OUTPUT :
103
104 ***** MAIN MENU *****
105 1. Insert an element
106 2. Delete an element
107 3. Peek
108 4. Display the queue
109 5. EXIT
110 Enter your option :
111 1
112 Enter the number to be inserted in the queue :
113 1
114 ***** MAIN MENU *****
115 1. Insert an element
116 2. Delete an element
117 3. Peek
118 4. Display the queue
119 5. EXIT
120 Enter your option :
121 1
122 Enter the number to be inserted in the queue :
123 2
124 ***** MAIN MENU *****
125 1. Insert an element
126 2. Delete an element
127 3. Peek
128 4. Display the queue
129 5. EXIT
130 Enter your option :
131 1
132 Enter the number to be inserted in the queue :
133 3
134 ***** MAIN MENU *****
135 1. Insert an element
136 2. Delete an element
137 3. Peek
138 4. Display the queue
139 5. EXIT
140 Enter your option :
141 1
142 Enter the number to be inserted in the queue :
143 4
144 ***** MAIN MENU *****
145 1. Insert an element
146 2. Delete an element
147 3. Peek
148 4. Display the queue
149 5. EXIT
150 Enter your option :
151 4
152
153 1      2      3      4      ***** MAIN MENU *****
154 1. Insert an element
155 2. Delete an element
156 3. Peek
157 4. Display the queue
158 5. EXIT
159 Enter your option :
160 2
161 The number deleted is : 1
162
163 ***** MAIN MENU *****
164 1. Insert an element
165 2. Delete an element
166 3. Peek
167 4. Display the queue
168 5. EXIT
169 Enter your option :
170 1
171 Enter the number to be inserted in the queue :
172 3
173 ***** MAIN MENU *****
174 1. Insert an element
175 2. Delete an element
176 3. Peek
177 4. Display the queue
178 5. EXIT
179 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("\n 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 :-

```
77  OUTPUT :
78  1: Insert
79  2:Delete
80  3: Display
81  4: Exit
82  Enter your choice
83  1
84  Enter the element:
85  2
86  1: Insert
87  2:Delete
88  3: Display
89  4: Exit
90  Enter your choice
91  1
92  Enter the element:
93  3
94  1: Insert
95  2:Delete
96  3: Display
97  4: Exit
98  Enter your choice
99  1
100 Enter the element:
101 4
102 1: Insert
103 2:Delete
104 3: Display
105 4: Exit
106 Enter your choice
107 1
108 Queue is full
109
110 1: Insert
111 2:Delete
112 3: Display
113 4: Exit
114 Enter your choice
115 3
116 2 3 4
117 1: Insert
118 2:Delete
119 3: Display
120 4: Exit
121 Enter your choice
122 2
123 2 is deleted
124 1: Insert
125 2:Delete
126 3: Display
127 4: Exit
128 Enter your choice
129 3
130 3 4
131 1: Insert
132 2:Delete
133 3: Display
134 4: Exit
135 Enter your choice
136
```


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;
}
```

OUTPUT :-

```
183  OUTPUT:
184
185  Enter the operation
186  1.Add in begin
187  2.Add at end
188  3.Add after a node
189  4.Delete node
190  5.Display
191  6.Length of list
192  7.Exit
193  1
194  Enter node element
195  1
196  Enter the operation
197  1.Add in begin
198  2.Add at end
199  3.Add after a node
200  4.Delete node
201  5.Display
202  6.Length of list
203  7.Exit
204  1
205  Enter node element
206  2
207  Enter the operation
208  1.Add in begin
209  2.Add at end
210  3.Add after a node
211  4.Delete node
212  5.Display
213  6.Length of list
214  7.Exit
215  1
216  Enter node element
217  3
218  Enter the operation
219  1.Add in begin
220  2.Add at end
221  3.Add after a node
222  4.Delete node
223  5.Display
224  6.Length of list
225  7.Exit
226  1
227  Enter node element
228  4
229  Enter the operation
230  1.Add in begin
231  2.Add at end
232  3.Add after a node
233  4.Delete node
234  5.Display
235  6.Length of list
236  7.Exit
237  1
238  Enter node element
239  5
240  Enter the operation
241  1.Add in begin
242  2.Add at end
243  3.Add after a node
244  4.Delete node
245  5.Display
246  6.Length of list
247  7.Exit
248  1
```



```
248 1
249 Enter node element
250 6
251 Enter the operation
252 1.Add in begin
253 2.Add at end
254 3.Add after a node
255 4.Delete node
256 5.Display
257 6.Length of list
258 7.Exit
259 5
260 6
261 5
262 4
263 3
264 2
265 1
266 Enter the operation
267 1.Add in begin
268 2.Add at end
269 3.Add after a node
270 4.Delete node
271 5.Display
272 6.Length of list
273 7.Exit
274 4
275 Enter the locatin of node to be deleted
276 3
277 Enter the operation
278 1.Add in begin
279 2.Add at end
280 3.Add after a node
281 4.Delete node
282 5.Display
283 6.Length of list
284 7.Exit
285 5
286 6
287 5
288 3
289 2
290 1
291 Enter the operation
292 1.Add in begin
293 2.Add at end
294 3.Add after a node
295 4.Delete node
296 5.Display
297 6.Length of list
298 7.Exit
299 1
300 Enter node element
301 2
302 Enter the operation
303 1.Add in begin
304 2.Add at end
305 3.Add after a node
306 4.Delete node
307 5.Display
308 6.Length of list
309 7.Exit
310 5
311 2
312 6
313 5
314 3
315 2
316 1
317 Enter the operation
```

```
310 5
311 2
312 6
313 5
314 3
315 2
316 1
317 Enter the operation
318 1.Add in begin
319 2.Add at end
320 3.Add after a node
321 4.Delete node
322 5.Display
323 6.Length of list
324 7.Exit
325 3
326 Enter the location1
327 Enter the node element
328 2
329 Enter the operation
330 1.Add in begin
331 2.Add at end
332 3.Add after a node
333 4.Delete node
334 5.Display
335 6.Length of list
336 7.Exit
337 5
338 2
339 2
340 6
341 5
342 3
343 2
344 1
345 Enter the operation
346 1.Add in begin
347 2.Add at end
348 3.Add after a node
349 4.Delete node
350 5.Display
351 6.Length of list
352 7.Exit
353 6
354 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:

```

2 *****MENU*****
3 Choose an option from the list:
4 -----STACK OPERATIONS-----
5 [1]PUSH
6 [2]POP
7 -----QUEUE OPERATIONS-----
8 [3]ENQUEUE
9 [4]DEQUEUE
10 -----
11 [5]DISPLAY
12 [6]SORT
13 [7]REVERSE
14 [8]CONCATENATION
15 [9]EXIT
16
17 Enter your choice:1
18
19 Enter the Value to be inserted:1
20
21 NODE INSERTED AT THE TOP OF THE STACK
22
23
24 *****MENU*****
25 Choose an option from the list:
26 -----STACK OPERATIONS-----
27 [1]PUSH
28 [2]POP
29 -----QUEUE OPERATIONS-----
30 [3]ENQUEUE
31 [4]DEQUEUE
32 -----
33 [5]DISPLAY
34 [6]SORT
35 [7]REVERSE
36 [8]CONCATENATION
37 [9]EXIT
38
39 Enter your choice:1
40
41 Enter the Value to be inserted:2
42
43 NODE INSERTED AT THE TOP OF THE STACK
44
45
46 *****MENU*****
47 Choose an option from the list:
48 -----STACK OPERATIONS-----
49 [1]PUSH
50 [2]POP
51 -----QUEUE OPERATIONS-----
52 [3]ENQUEUE
53 [4]DEQUEUE
54 -----
55 [5]DISPLAY
56 [6]SORT
57 [7]REVERSE
58 [8]CONCATENATION
59 [9]EXIT
60
61 Enter your choice:1
62
63 Enter the Value to be inserted:4
64
65 NODE INSERTED AT THE TOP OF THE STACK
66
67
68 *****MENU*****
69 Choose an option from the list:
70 -----STACK OPERATIONS-----
71 [1]PUSH
72 [2]POP
73 -----QUEUE OPERATIONS-----
74 [3]ENQUEUE
75 [4]DEQUEUE
76 -----
77 [5]DISPLAY
78 [6]SORT
79 [7]REVERSE
80 [8]CONCATENATION
81 [9]EXIT
82
83 Enter your choice:1
84

```

```

83 Enter your choice:1
84
85 Enter the Value to be inserted:3
86
87 NODE INSERTED AT THE TOP OF THE STACK
88
89
90 *****MENU*****
91 Choose an option from the list:
92 -----STACK OPERATIONS-----
93 [1]PUSH
94 [2]POP
95 -----QUEUE OPERATIONS-----
96 [3]ENQUEUE
97 [4]DEQUEUE
98 -----
99 [5]DISPLAY
100 [6]SORT
101 [7]REVERSE
102 [8]CONCATENATION
103 [9]EXIT
104
105 Enter your choice:5
106
107
108 LIST--> 3      4      2      1
109
110 *****MENU*****
111 Choose an option from the list:
112 -----STACK OPERATIONS-----
113 [1]PUSH
114 [2]POP
115 -----QUEUE OPERATIONS-----
116 [3]ENQUEUE
117 [4]DEQUEUE
118 -----
119 [5]DISPLAY
120 [6]SORT
121 [7]REVERSE
122 [8]CONCATENATION
123 [9]EXIT
124
125 Enter your choice:6
126
127 Sorted List::
128
129 LIST--> 1      2      3      4
130
131 *****MENU*****
132 Choose an option from the list:
133 -----STACK OPERATIONS-----
134 [1]PUSH
135 [2]POP
136 -----QUEUE OPERATIONS-----
137 [3]ENQUEUE
138 [4]DEQUEUE
139 -----
140 [5]DISPLAY
141 [6]SORT
142 [7]REVERSE
143 [8]CONCATENATION
144 [9]EXIT
145
146 Enter your choice:7
147
148 Reversed List::
149
150 LIST--> 4      3      2      1
151
152 *****MENU*****
153 Choose an option from the list:
154 -----STACK OPERATIONS-----
155 [1]PUSH
156 [2]POP
157 -----QUEUE OPERATIONS-----
158 [3]ENQUEUE
159 [4]DEQUEUE
160 -----
161 [5]DISPLAY
162 [6]SORT
163 [7]REVERSE
164 [8]CONCATENATION
165 [9]EXIT
166

```

```

152 *****MENU*****
153 Choose an option from the list:
154 -----STACK OPERATIONS-----
155 [1]PUSH
156 [2]POP
157 -----QUEUE OPERATIONS-----
158 [3]ENQUEUE
159 [4]DEQUEUE
160 -----
161 [5]DISPLAY
162 [6]SORT
163 [7]REVERSE
164 [8]CONCATENATION
165 [9]EXIT
166
167 Enter your choice:8
168
169 Create a Second list-->
170 Enter the number of nodes : 2
171 Enter the element to be inserted : 5
172 Enter the element to be inserted : 6
173
174 List1:
175
176 LIST--> 4      3      2      1
177
178 LIST2-->      5      6
179
180 Concatenated List:
181
182 LIST--> 4      3      2      1      5      6
183
184 *****MENU*****
185 Choose an option from the list:
186 -----STACK OPERATIONS-----
187 [1]PUSH
188 [2]POP
189 -----QUEUE OPERATIONS-----
190 [3]ENQUEUE
191 [4]DEQUEUE
192 -----
193 [5]DISPLAY
194 [6]SORT
195 [7]REVERSE
196 [8]CONCATENATION
197 [9]EXIT
198
199 Enter your choice:6
200
201 Sorted List::
202
203 LIST--> 1      2      3      4      5      6
204
205 *****MENU*****
206 Choose an option from the list:
207 -----STACK OPERATIONS-----
208 [1]PUSH
209 [2]POP
210 -----QUEUE OPERATIONS-----
211 [3]ENQUEUE
212 [4]DEQUEUE
213 -----
214 [5]DISPLAY
215 [6]SORT
216 [7]REVERSE
217 [8]CONCATENATION
218 [9]EXIT
219
220 Enter your choice:2
221
222 NODE DELETED FROM TOP OF THE STACK
223
224
225 *****MENU*****
226 Choose an option from the list:
227 -----STACK OPERATIONS-----
228 [1]PUSH
229 [2]POP
230 -----QUEUE OPERATIONS-----
231 [3]ENQUEUE
232 [4]DEQUEUE
233 -----
234 [5]DISPLAY
235

```



```

243 LIST--> 2      3      4      5      6
244
245 *****MENU*****
246 Choose an option from the list:
247 -----STACK OPERATIONS-----
248 [1]PUSH
249 [2]POP
250 -----QUEUE OPERATIONS-----
251 [3]ENQUEUE
252 [4]DEQUEUE
253 -----
254 [5]DISPLAY
255 [6]SORT
256 [7]REVERSE
257 [8]CONCATENATION
258 [9]EXIT
259
260 Enter your choice:3
261
262 Enter the Value to be inserted:1
263
264 NODE INSERTED AT REAR OF THE QUEUE
265
266 *****MENU*****
267 Choose an option from the list:
268 -----STACK OPERATIONS-----
269 [1]PUSH
270 [2]POP
271 -----QUEUE OPERATIONS-----
272 [3]ENQUEUE
273 [4]DEQUEUE
274 -----
275 [5]DISPLAY
276 [6]SORT
277 [7]REVERSE
278 [8]CONCATENATION
279 [9]EXIT
280
281 Enter your choice:5
282
283
284
285 LIST--> 2      3      4      5      6      1
286
287 *****MENU*****
288 Choose an option from the list:
289 -----STACK OPERATIONS-----
290 [1]PUSH
291 [2]POP
292 -----QUEUE OPERATIONS-----
293 [3]ENQUEUE
294 [4]DEQUEUE
295 -----
296 [5]DISPLAY
297 [6]SORT
298 [7]REVERSE
299 [8]CONCATENATION
300 [9]EXIT
301
302 Enter your choice:
303 6
304
305 Sorted List::
306
307 LIST--> 1      2      3      4      5      6
308
309 *****MENU*****
310 Choose an option from the list:
311 -----STACK OPERATIONS-----
312 [1]PUSH
313 [2]POP
314 -----QUEUE OPERATIONS-----
315 [3]ENQUEUE
316 [4]DEQUEUE
317 -----
318 [5]DISPLAY
319 [6]SORT
320 [7]REVERSE
321 [8]CONCATENATION
322 [9]EXIT
323
324 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 :

```

1
2 Sir output was a bit long, which required multiple screenshots, so i have copied and pasted the output of output contents
3
4
5 *****MENU*****
6 Choose an option from the list:
7 [1]CREATE A LIST
8 [2]INSERT TO THE LEFT OF A NODE
9 [3]DELETE NODE
10 [4]DISPLAY
11 [5]EXIT
12
13 Enter your choice:1
14
15 Enter the number of nodes:4
16
17 Enter the value to be inserted for Node 1 :    1
18 Enter the value to be inserted for Node 2 :    2
19 Enter the value to be inserted for Node 3 :    3
20 Enter the value to be inserted for Node 4 :    4
21
22
23 Linked List Created!!
24
25 *****MENU*****
26 Choose an option from the list:
27 [1]CREATE A LIST
28 [2]INSERT TO THE LEFT OF A NODE
29 [3]DELETE NODE
30 [4]DISPLAY
31 [5]EXIT
32
33 Enter your choice:3
34
35 Enter the Value to be deleted: 2
36 Value 2, deleted
37
38
39 *****MENU*****
40 Choose an option from the list:
41 [1]CREATE A LIST
42 [2]INSERT TO THE LEFT OF A NODE
43 [3]DELETE NODE
44 [4]DISPLAY
45 [5]EXIT
46
47 Enter your choice:2
48
49 Enter the Node to insert the value: 1
50
51 Enter the value to be inserted: 2
52
53
54 Node Inserted at 1 position!!
55
56 *****MENU*****
57 Choose an option from the list:
58 [1]CREATE A LIST
59 [2]INSERT TO THE LEFT OF A NODE
60 [3]DELETE NODE
61 [4]DISPLAY
62 [5]EXIT
63
64 Enter your choice:4
65
66
67 LIST--> 2      1      3      4
68
69 *****MENU*****
70 Choose an option from the list:
71 [1]CREATE A LIST
72 [2]INSERT TO THE LEFT OF A NODE
73 [3]DELETE NODE
74 [4]DISPLAY
75 [5]EXIT
76
77 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");
```

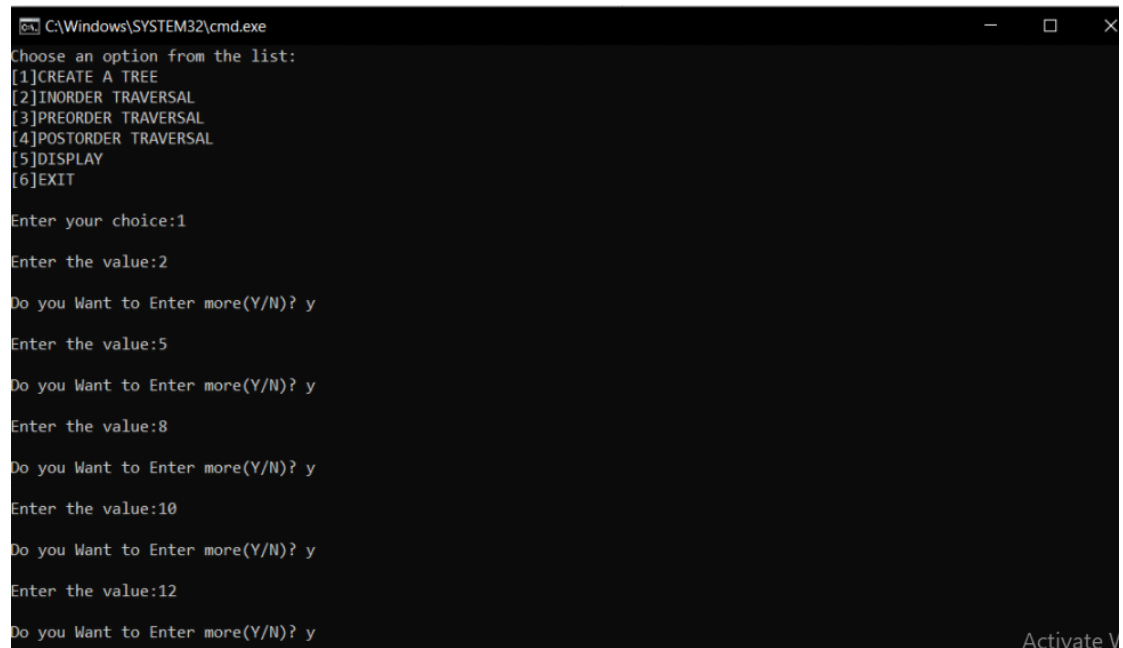
```
}
```

```
}
```

```
return 0;
```

```
}
```

OUTPUT :-



```
C:\Windows\SYSTEM32\cmd.exe
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:1

Enter the value:2

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

Enter the value:5

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

Enter the value:8

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

Enter the value:10

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

Enter the value:12

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

Activate Windows
```

```

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

```

