

PROGRAM - 1

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

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

```
#define SIZE 100
```

```
void push(int);
```

```
void pop();
```

```
void display();
```

```
int stack[SIZE], top = -1;
```

```
void push(int value) {
```

```
    if (top == SIZE - 1)
```

```
        printf("\n Stack is Full");
```

```
    else {
```

```
        top = top + 1;
```

```
        stack[top] = value;
```

```
    }
```

```
}
```

```
void pop() {
```

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

```
        printf("\n stack is empty");
```

```
    else {
```

```
        printf("\n Deleted : %d", stack[top]);
```

```
        top = top - 1;
```

```
}
```

```
}
```

Handwritten notes:
Printed
11/2/2024


```
void display () {
```

```
if (top == -1)
```

```
    Print ("\n stack is empty!");
```

```
else {
```

```
    int i;
```

```
    printf ("\n stack elements are: \n");
```

```
    for (i = top; i >= 0; i--)
```

```
        printf ("%d \n", stack[i]);
```

```
}
```

```
}
```

```
void main () {
```

```
int value, choice;
```

```
while (1) {
```

```
    printf ("\n\n *** MENU *** \n");
```

```
    printf ("1. Push \n, 2. Pop \n 3. Display \n 4. Exit");
```

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

```
    switch (choice) {
```

```
case 1: printf ("Enter the value to be inserted:");
```

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

```
        push (value);
```

```
        break;
```

```
case 2: pop ();
```

```
        break;
```

```
case 3: display ();
```

```
        break;
```

```
case 4: exit (0);
```

```
default: printf ("\n ERROR \n");
```

```
}
```

```
}
```

```
}
```


OUTPUT:

*** MENU ***

1. PUSH

2. POP

3. Display

4. Exit

1

Enter value to be inserted : 2

** MENU **

1. PUSH

2. POP

3. DISPLAY

4. Exit

2

Enter value to be inserted : 2

** MENU **

:

:

3

Stack Elements are: 1 2

2

1

** MENU **

:

:

2

Deleted : 2

** MENU **

:

4

Program 2: (Infix to postfix)

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 30
char Stack[SIZE];
int top = -1;

void Push (char ele)
{
    stack[++top] = ele;
}

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, ele;
    int i = 0, k = 0;
    printf("Enter infix expression: ");
    scanf("%s", infix);
    Push("#");
```


while ((ch = infix[i++]) != "\0")
{

if (ch == 'c') {

Push(ch);

else if (isalnum(ch)) {

Postfix[k++] = ch; }

else if (ch == '(') {

while (stack[top] != 'c') {

Postfix[k++] = pop(); }

ch = pop();

}

else {

while (pr(stack[top]) >= pr(ch)) {

Postfix[k++] = pop(); }

Push(ch);

}

}

while (stack[top] != '#') {

Postfix[k++] = pop(); }

Postfix[k] = '\0';

Printf("The postfix expression is: %s", postfix);

}

Output

Enter infix expression: a+b+c

The postfix expression is: abc++

linear Queue

Program 3:

```

#include <stdio.h>
#include <conio.h>
#include <process.h>
#define SIZE 5
int item, front = 0, rear = -1, q[10];
void insert()
{
    if (rear == SIZE - 1)
    {
        printf("queue overflow\n");
        return;
    }
    rear = rear + 1;
    q[rear] = item;
}
int delete() {
    if (front > rear)
        return -1;
    return q[front++];
}
void display() {
    int i;
    if (front > rear) {
        printf("Queue is empty");
        return;
    }
    printf("contents of queue\n:");
    for (i = front; i < rear; i++) {
        printf("%d\n", q[i]);
    }
}

```



```

void main () {
    int choice;
    while (1) {
        printf("\n 1: insert \n 2: delete \n 3: display \n 4: exit\n");
        printf("Enter choice\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1: printf("Enter the item to be inserted\n");
                    scanf("%d", &item);
                    insertrear();
                    break;
            case 2: item = deletefront();
                    if (item == -2)
                        printf("Queue is empty\n");
                    else
                        printf("item deleted = %d\n", item);
                    break;
            case 3: display();
                    break;
            default: exit(0);
        }
    }
}

```


Program 4 : Circular queues

```

#include <stdio.h>
#include <conio.h>
#include <process.h>
#define SIZE 5
int item, front = 0, rear = -1, q[10];
void insert_rear (int item, int *rear, int q[])
{
    if (*rear == SIZE - 1)
    {
        printf("Overflow");
        return;
    }
    *rear = *rear + 1;
    q[*rear] = item;
}

int delete_front (int *front, int *rear, int q[])
void insert_rear() {
    if (count == SIZE)
    {
        printf("Queue Overflow");
        return;
    }
    rear = (rear + 1) % SIZE;
    q[rear] = item;
    count++;
}

int delete_front()
{
    if (count == 0) return -1;
    item = q[front];
    front = (front + 1) % SIZE;
    count--;
}

```



```

return item;
}

void display () {
int i, f;
if (count == 0)
{
printf("queue is empty");
return;
}
f = front;
printf("contents of queue\n");
for (i = 1; i <= count; i++)
{
printf("%d\n", q[f]);
f = (f + 1) % SIZE;
}
}

void main () {
int choice;
while (1) {
printf("\n 1: insert\n 2: delete\n 3: display\n 4: exit");
scanf("%d", &choice);
switch (choice)
{
case 1: printf("Enter the item to be inserted\n");
scanf("%d", &item);
insertrear();
break;
case 2: item = deletefront();
if (item == -1) {
printf("Empty");
}
}
}
}

```



```
else {  
    printf("deleted item is: %d", item);  
    break;  
case 3: display();  
    break;  
default: exit(0);
```

Both
0/18hr

3
3
3

9/2/24
8/1/24