PRIORITY QUEUE

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int pqueue[MAX];
int front = rear = -1;
void enqueue (int item)
{
    if(rear >= MAX - 1)
    {
        printf("\n QUEUE OVERFLOW");
        return;
    }
    if ((front == -1) && (rear == -1))
    {
        front = rear = 0;
        pqueue [rear] = item;
        return;
    }
    else
        check_priority (item);
    rear++;
}
void check_priority (int item)
{
    int i,j;
    for(i=0 ; i<=rear; i++)
    {
        if(item >= pqueue[i])
        {
            for(j=rear+1; j>i; j--)
            {
                pqueue[j] = pqueue[j-1];
            }
            pqueue[i] = item;
            return;
        }
    }
    pqueue[i] = item;
}
void dequeue (int item)
{
    int i;
    if ((front == -1) && (rear == -1))
    {
        printf("\nEmpty Queue");
        return;
    }
    for(i=0; i<=rear; i++)
    {
        if (item == pqueue[i])
        {
            for( ; i<rear; i++)
            {
                pqueue[i] = pqueue[i+1];
            }
            pqueue[i] = -99;
            rear--;
            if(rear == -1)
                front = -1;
            return;
        }
    }
    printf("\n %d element not found in queue", item);
}
void display()
{
    if ((front == -1) && (rear == -1))
    {
        printf("\n Empty Queue");
        return;
    }
    for( ; front <= rear; front++)
    {
        printf(" %d ", pqueue[front]);
    }
    front = 0;
}
void main ()
{
    int n, choice;
    printf("\n Enter 1 to insert element by priority");
    printf("\n Enter 2 to delete element by priority");
    printf("\n Enter 3 to display priority queue");
    printf("\n Enter 4 to exit");
    while(1)
    {
        printf("\n Enter your choice : ");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1 : printf("Enter element to insert : ");
                     scanf("%d", &n);
                     enqueue(n);
                     break;
            case 2 : printf("Enter element to delete: ");
                     scanf("%d", &n);
                     dequeue(n);
                     break;
            case 3 : display();
                     break;
            case 4 : exit(0);
            default : printf("\n Please enter valid choice");
        }
    }
}
```

INFIX TO POSTFIX

```
#include <stdio.h>
#include <ctype.h>
char stack[100];
int top=-1;
void push(char x)
{
    stack[++top]=x;
}
char pop()
{
    if(top==-1)
        return -1;
    else
        return stack[top--];
}
int priority(char x)
{
    if(x=='(')
        return 0;
    if(x=='+'||x=='-')
        return 1;
    if(x=='*'||x=='/')
        return 2;
    return 0;
}
int main()
{
    char exp[100];
    char *e, x;
    printf("Enter the expression: ");
    scanf("%s",exp);
    printf("\n");
    e=exp;
    while(*e!='\0')
    {
        if(isanum(*e))
            printf("%c",*e);
        else if(*e=='(')
            push(*e);
        else if(*e==')')
        {
            while((x=pop())!='(')
                printf("%c",x);
        }
        else
        {
            while(priority(stack[top])>=
                        priority(*e))
                printf("%c",pop());
            push(*e);
        }
        e++;
    }
    while(top!=-1)
    {
        printf("%c",pop());
    }
    return 0;
}
```

EVALUATION OF POSTFIX

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#define SIZE 40

char postfix[SIZE];
int stack[SIZE], top=-1;
int main()
{
    int i,a,b, result, pEval;
    char ch;
    for(i=0; i<SIZE; i++)
    {
        stack[i]=-1;
    }
    printf("Enter a postfix expression: ");
    scanf("%s", postfix);
    for(i=0; postfix[i]!='\0'; i++)
    {
        ch = postfix[i];
        if(isdigit(ch))
        {
            push(ch-'0');
        }
        else if(ch=='+' || ch=='-' || ch=='*' || ch=='/')
        {
            b=pop();
            a=pop();
            switch(ch)
            {
                case '+': result = a+b;
                          break;
                case '-': result= a-b;
                          break;
                case '*': result= a*b;
                          break;
                case '/': result= a/b;
                          break;
                case '%': result= a%b;
                          break;
            }
            push(result);
        }
    }
    pEval = pop();
    printf("\n The postfix evaluation is: %d\n", pEval);
    return 0;
}

void push(int n)
{
    if(top<SIZE-1)
    {
        stack[++top]=n;
    }
    else
    {
        printf("Stack is full!\n");
        exit(-1);
    }
}
int pop()
{
    int n;
    if(top>-1)
    {
        n=stack[top];
        stack[top--]=-1;
        return n;
    }
    else
    {
        printf("Stack is empty!\n");
        exit(-1);
    }
}
```