Evaluation of Postfix Expression

```
//Evaluation of Postfix expression

#include<stdio.h>
#include<math.h>
void push(char b);
void push1(int n);
char pop();
int pop1();
int icp(char p);
int isp(char q);
char a[100],a1[100],out[20],e[20];
int top=-1,max=20;
main()
{
  char item,op,x,y;
  int t,value,num;
  int i=0,c=0,j,p,q;
  printf("\nEnter an infix expression\t");
  scanf("%s",e);
  for(j=0;e[j]!='\0';j++);
  e[j]=')';
  push('(');
  while(top>-1)
    {
      item=e[i++];
      if(item=='\0')
  {
    break;
  }
      else if(isalnum(item))
  {
    out[c]=item;
    c++;
  }
      else if(item==')')
  {
    x=pop();
    while(x!='(')
      {
        out[c]=x;
        c++;
        x=pop();
      }
  }
      else if(isp(a[top])>=icp(item))
  {
```

```
    x=pop();
    while(isp(x)>=icp(item))
      {
        out[c]=x;
        c++;
        x=pop();
      }
    push(x);
    push(item);
}
      else if(isp(a[top])<icp(item))
{
    push(item);
}
      else
printf("invalid expression");
    }
top=-1;
for(j=0;out[j]!='\0';j++);
out[j]='#';
out[j+1]='\0';
i=0;
while(item!='#')
    {
      item=out[i];
      if(isalnum(item))
{
    printf("Enter the value of %c\t",item);
    scanf("%d",&num);
    push1(num);
}
      else
{
    op=item;
    p=pop1();
    q=pop1();
    switch(op)
      {
      case '+':
        t=q+p;
        break;
      case '-':
        t=q-p;
        break;
      case '^':
        t=pow(q,p);
        break;
      case '*':
```

```
        t=q*p;
        break;
      case '/':
        t=q/p;
        break;
      }
    push1(t);
  }
      i++;
    }
  value=pop1();
  printf("\nThe value is\t");
  printf("%d\n",value);
}

void push(char b)
{
  if(top>=(max-1))
    {
      printf("stack overflow\n");
    }
  else
    {
      top=top+1;
      a[top]=b;
    }
}

void push1(int n)
{
  if(top>=(max-1))
    {
      printf("stack overflow\n");
    }
  else
    {
      top=top+1;
      a1[top]=n;
    }
}

char pop()
{
  int d;
  if(top<0)
    {
      printf("the stack is empty\n");
    }
```

```
  else
    {
      d=a[top];
      top=top-1;
    }
  return d;
}

int pop1()
{
  int d;
  if(top>=0)
    {
      d=a1[top];
      top=top-1;
    }
  return d;
}

int isp(char q)
{
  if(q=='^')
    return 3;
  else if(q=='*')
    return 2;
  else if(q=='/')
    return 2;
  else if(q=='+')
    return 1;
  else if(q=='-')
    return 1;
  else if(q=='(')
    return 0;
}

int icp(char p)
{
  if(p=='^')
    return 4;
  else if(p=='*')
    return 2;
  else if(p=='/')
    return 2;
  else if(p=='+')
    return 1;
  else if(p=='-')
    return 1;
  else if(p=='(')
```

```
    return 4;
}
```

Output

```
Enter an infix expression (a^b)/(c*d)
Enter the value of a  2
Enter the value of b  4
Enter the value of c  8
Enter the value of d  1

The value is  2
```

Array Implementation of Linear Queue

```
//Array Implementation of Linear Queue
#include<stdio.h>
void insert();
void delete();
void display();
int rear=0,front=0,size;
int queue[20];
main()
{
  int i,choice;
  printf("\nEnter the Size of Queue: ");
  scanf("%d",&size);
  do
    {
      printf("\n\n\nMenu\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\n");
      printf("Enter your choice ");
      scanf("%d",&choice);
      switch(choice)
  {
  case 1:
    insert();
    break;
  case 2:
    delete();
    break;
  case 3:
    display();
    break;
  case 4:
    exit(0);
  default:
    printf("\nInvalid entry");
  }
    }while(choice!=4);
}

void insert()
{
  int i;
  if(rear==size)
    {
      printf("\nQueue is full\n");
    }
  else
    {
      for(i=0;i<1;i++)
```

```
  {
    rear=rear+1;
    printf("\nEnter the element to be inserted: ");
    scanf("%d",&queue[rear]);
  }
    }
  if(front==0)
    {
      front=1;
    }
}
void delete()
{
  int item;
  if(front==0)
    {
      printf("\nQueue is empty\n");
    }
  else
    {
      item=queue[front];
      printf("\nThe deleted element is %d",item);
      if(front==rear)
  {
    front=0;
    rear=0;
  }
      else
  {
    front=front+1;
  }
    }
}
void display()
{
  int i;
  if(front==0)
    {
      printf("\nQueue is empty\n");
    }
  else
    {
      printf("\nThe queue is\n");
      for(i=front;i<=rear;i++)
        {
          printf(" %d ",queue[i]);
        }
    }
```

```
}
```

Output

```
Enter the Size of Queue: 5
Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 1
Enter the element to be inserted: 6

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 1
Enter the element to be inserted: 7

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 1
Enter the element to be inserted: 8

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 1
Enter the element to be inserted: 9

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 3
The queue is
 6  7  8  9

Menu
1.Enqueue
2.Dequeue
```

```
3.Display
4.Exit
Enter your choice 2
The deleted element is 6

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 2
The deleted element is 7

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 1
Enter the element to be inserted: 5

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 3
The queue is
 8  9  5

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice 4
```

Implementation of Circular Queue

```
//Implementation of Circular Queue
#include<stdio.h>
void insert();
void delete();
void display();
int rear=0,front=0,size;
int cqueue[5];
main()
{
  printf("\nEnter the Size of the circular Queue : ");
  scanf("%d",&size);
  int i,choice;
  do
    {
      printf("\nMenu\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&choice);
      switch(choice)
  {
  case 1:
    insert();
    break;
  case 2:
    delete();
    break;
  case 3:
    display();
    break;
  case 4:
    exit(0);
  default:
    printf("\nInvalid entry");
  }
    }while(choice!=4);
}

void insert()
{
  int i;
  if(((rear%size)+1)==front)
    {
      printf("\nCircular Queue is full\n");
    }
  else
    {
      for(i=0;i<1;i++)
```

```
  {
    rear=((rear%size)+1);
    printf("\nEnter the element to be inserted: ");
    scanf("%d",&cqueue[rear]);
  }
    }
  if(front==0)
    {
      front=1;
    }
}
void delete()
{
  int item;
  if(front==0)
    {
      printf("\nCircular Queue is empty\n");
    }
  else
    {
      item=cqueue[front];
      printf("\nThe deleted element is %d",item);
      if(front==rear)
  {
    front=0;
    rear=0;
  }
      else
  {
    front=(front%size)+1;
  }
    }
}
void display()
{
  int i;
  if(front==0)
    {
      printf("\nCircular Queue is empty\n");
    }
  else
    {
      printf("\nThe circular queue is\n");
      for(i=front;i!=rear;i=((i%size)+1))
        {
          printf(" %d ",cqueue[i]);
        }
      printf(" %d ",cqueue[rear]);
```

```
    }
}
```

Output

```
Enter the Size of the circular Queue : 5
Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1
Enter the element to be inserted: 1

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1
Enter the element to be inserted: 2

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1
Enter the element to be inserted: 3

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1
Enter the element to be inserted: 4

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1
Enter the element to be inserted: 5

Menu
1.Enqueue
2.Dequeue
```

```
3.Display
4.Exit
Enter your choice: 1
Circular Queue is full

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 3
The circular queue is
 1  2  3  4  5

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 2
The deleted element is 1

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 2
The deleted element is 2

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 3
The circular queue is
 3  4  5

Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 4
```