

## INFIX TO POSTFIX

void push (char c)

```
{  
    if (top == MAX-1)  
        printf ("stack overflow");  
    else  
    {  
        top++;  
        stack[top] = c;  
    }  
}
```

char pop ()

```
{  
    char e;  
    if (top == -1)  
        return -1;  
    else  
    {  
        e = stack[top];  
        top--;  
        return e;  
    }  
}
```

}

int isoperator (char c)

```
{  
    if (c == '^' || c == '*' || c == '/' || c == '+' || c == '-')  
        return 1;  
    else  
        return 0;  
}
```

int precedence (char c)

{

if (c == '(')

return 3;

else if (c == '\*' || c == '/')

return 2;

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

return 1;

else return 0;

void convert (char infix[], char postfix[])

{

int i = 0, j = 0;

char item, x;

push('(');

strcat(infix, "(");

item = infix[j];

while (item != '\0')

{

if (item == '(')

push(item);

else if (!isdigit(item) || isalpha(item))

{

postfix[j] = item;

j++;

}

else if (isoperator(item) == 1)

{

x = pop();

while (isoperator(x) == 1 && precedence(x) >= precedence(item))

{

postfix[j] = x;

j++;

x = pop();

}

push(x);

push(item);

}

else if (item == ')')

{

x = pop();

while (item != '(')

{

postfix[j] = x;

j++;

x = pop();

}

}

else

```
{ printf ("Invalid Expression!\n");  
  exit (0);
```

```
}
```

```
  i++;
```

```
  item = infix[i];
```

```
}
```

```
if (top > 0)
```

```
{
```

```
  printf ("Invalid expression!!!\n");
```

```
  exit (0);
```

```
}
```

```
postfix[j] = '\0';
```

```
}
```