B] Infix to postfix Conversion

```
include <iostream>
Using namespace std;
int priority (char c);
char infix[20], stack[20];

int I = 0, j = 0;
int Main ()
{
int p1;
cout << “Enter infix expression:”;
cin >> infix;
cout << “output expression is:”;

  do
   {
p1 = priority (infix[i]);
if (p1 == 6)
cout << infix[i];
     else
           {
if (j == 0)
              {
stack[j] = infix[i];
j++;
}
     else
if (p1 == 4)
              {
```

```
stack[j] = infix[i];
j++;
}
      else
if (p1 == 5)
{
              do
                      {
cout << stack[j – 1] << " ";
j--;
}
                while (stack[j – 1] != '(');
       j--;
}
           else
if (p1 > priority (stack[j – 1]) || stack[j – 1] == '('
              {
stack[j] = infix[i];
j++;
}
             else
if (p1 <= priority (stack[j – 1]))
              {
while (p1 <= priority (stack[j – 1]) && stack[j – 1] != '('
                          && j != 0)
                      {
cout << stack[j – 1] << " ";
j--;
```

```
}
stack[j] = infix[i];
j++;
}                                   //end of if
            }                                   //end of main else
     i++;
}
  while (infix[i] != NULL);
for (I = j – 1; I >= 0; i--)    //print remaining operators of stack
    cout << stack[i] << " ";
return 0;
}
int
priority (char c)
{
int p;
iIf (c == '+' || c == '-')
P = 1;
  else
if (c == '*' || c == '/')
P = 2;
  else
if (c == '(')
p = 4;
  else
if (c == ')')
p = 5;
  else
p = 6;
```

```
return (p);
}
```

Output:

```
Enter infix expression:a+b
Output expression is:ab+
Enter infix expression:a+b*c
Output expression is:abc* +
Enter infix expression☹a+b)*(c-d)
Output expression is:ab+cd-*
```