1BM19CS191

```
ans = ans + s.top();
s.pop();
}
return ans;
}
```

Aditya Satish Kumar
1BM19CS191

DATE: PAGE:

Q WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operations + (plus), - (minus), * (multiply) and / (divide).

Ans

```
{ Create a stack S.
for i=0 to length (exp)-1.
{
 If ( exp[i] is ~~expand~~ operand
      res = res + exp[i];
 else if exp[i] is operator
 {
   while (!s.empty() && has higher precedence (s.top(), exp[i]))
   {
      res = res + s.top();
      s.pop();
   }
   s.push(exp[i]);
 }}
else if opening parenthesis (exp[i])
      s.push(exp[i]);
else if closing parenthesis (exp[i])
while (!s.empty && ! is open parenthesis.s(top));
{
    res = res + s.top();
    s.pop();
}
s.pop();
while (!s.empty())
{
```

Aditya Satish Kumar
1BM19CS191

DATE:
PAGE:

8. WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operations + (plus), - (minus), * (multiply) and / (divide)

Ans

```
{ Create a stack S.
for i=0 to length (exp)-1.
{
 If exp[i] is operand
        res = res + exp[i];
 else if exp[i] is operator
 {
   while (!S.empty() && has higher precidency (S.top(), exp[i]))
   {
        res = res + S.top();
        S.pop();
   }
   S.push(exp[i]);
} }
else if opening parenthesis (exp[i])
        S.push(exp[i]);
else if closing parenthesis (exp[i])
while (!S.empty && ! is open parenthesis. S(top));
{
    res = res + S.top();
    S.pop();
}
S.pop();
while (!S.empty())
{
```

1BM19CS191

```
    ses = ses + s.top();
    s.pop();
    }
    return ses;
}
```