```
#include <iostream>
using namespace std;
class stack
{
    char exp[20], s[20], s1[20];
    int top, max;
  public:
    stack()
    {  top = -1;
       max = 20;
    }
    char gettop();
    bool isfull();
    void expr();
    bool isempty();
    void push(char);
    char pop();
    void check(stack);
};
bool stack :: isempty()
{  if (top == -1)
    return 1;
    else
      return 0;
}
bool stack :: isfull()
{   if (top == max-1)
       return 1;
    else
       return 0;
}
```

```
char stack :: gettop()
{  if ( !isempty ())
      return s[top];
}
void stack :: push (char ex)
{  if (isfull())
       cout << " Stack overflow";
    else {
        s[++top] = ex;
       }
}
char stack :: pop()
{ if (isempty())
     cout << "Stack under flow ";
   else
      return s[top--];
}
void stack :: expr()
{   cout << " Enter the expression :: -> ";
    cin >> exp;
    cout << exp <<"\n";
}
void stack :: check (stack s)
{  char ex, ch;
    stack e;
    bool flag=0;
    for (int i=0; s.exp[i] != '\0'; i++)
    {  ex = s.exp[i];
       if (ex == '(' || ex == '{' || ex == '[')
```

```
{   e.push(ex);
    cout << "inserted :: "<<ex<<"\n";
}
else {
    ch=e.gettop();
    switch (ex)
    {   case ')':
            if (ch=='(')
            {  e.pop();
               flag=1;
            }
            else
               flag=0;
            break;
        case ']':
          if (ch=='[')
           {  e.pop();
              flag=1;
           }
           else
              flag=0;
           break;
        case '}':
          if (ch=='{')
          {  e.pop();
             flag=1;
          }
          else
             flag=0;
```

```
                break;
            }
        }
}
if (e.isempty() && flag==1)
{   cout << " expression is well parenthesized ";
}
else
    cout << " expression is well parenthesized ";
}
int main()
{   stack s;
    s.expr();
    s.check(s);
}
```

Output :-

```
Enter the Expression :: -> ({[]})
({[]})
insreted ::(
insreted ::{
insreted ::[
expression is well parenthesized.
```