```cpp
#include <iostream>
using  namespace  std  ;

struct  elemenet
{
    char               data  ;
    struct elemenet *  next  ;
};

typedef  struct elemenet Element  ;
typedef  Element *       Stack    ;

Stack  create_stack()
{
    return  NULL  ;
}

bool  empty_stack(Stack stk)
{
    if(stk == NULL)
        return  true  ;
    else
        return  false ;
}

int  depth_stack(Stack stk)
{
    Stack  ptr    =  stk  ;
    int    count  =  0    ;

    while(ptr != NULL)
    {
        count++  ;
        ptr  =  ptr->next  ;
    }

    return  count  ;
}

Stack  push(Stack stk, int val)
{
    Stack  ptr  =  new  Element  ;
    ptr->data   =  val  ;

    ptr->next  =  stk  ;
    stk        =  ptr  ;

    return  stk  ;
}

Stack  pop(Stack stk, int &val)
{
    if(!empty_stack(stk))
    {
        Stack  ptr  =  stk  ;

        val  =  stk->data  ;
        stk  =  stk->next  ;
        delete  ptr        ;

        return  stk  ;
    }
    else
    {
        cout  <<  "Error, the stack is already empty."  <<  endl  ;
        return  NULL  ;
    }
}

int  top_stack(Stack stk)
{
    if(empty_stack(stk))
    {
        cout  <<  "Error, Stack is empty."  <<  endl  ;
        return  0  ;
    }

```

```
78      return  stk->data  ;
79  }
80
81  Stack  replace_top_stack(Stack stk, int val)
82  {
83      if(empty_stack(stk))
84      {
85          cout  <<  "Error, Stack is empty."  <<  endl  ;
86          return  NULL  ;
87      }
88
89      stk->data  =  val  ;
90
91      return  stk  ;
92  }
93
94  void  display_stack(Stack stk)
95  {
96      Stack  ptr    =  stk  ;
97
98      if(empty_stack(stk))
99      {
100         cout  <<  "Stack is empty."  <<  endl  ;
101         return  ;
102     }
103
104     while(ptr != NULL)
105     {
106         cout  <<  ptr->data  <<  " "  ;
107         ptr  =  ptr->next  ;
108     }
109     cout  <<  endl  ;
110
111     return  ;
112 }
113
114 bool  parentheses_match(char * str)
115 {
116     Stack  stk  =  create_stack()  ;
117     int    val  ;
118
119     int  i = 0  ;
120     while(str[i] != '\0')
121     {
122         if((str[i] == '('||(str[i] == ')'||(str[i] == '['||(str[i] == ']')
              ||(str[i] == '{'||(str[i] == '}'))
123         {
124             if((str[i] == '('||(str[i] == '{'||(str[i] == '['))
125             {
126                 stk  =  push(stk, str[i])  ;
127             }
128             else
129             {
130                 if(empty_stack(stk))
131                 {
132                     return  false  ;
133                 }
134                 else
135                 {
136                     if(((str[i] == ')')&&(top_stack(stk) == '('))||((str[i] == '}')
                          && (top_stack(stk) == '{'))||((str[i] == ']')&&(top_stack(stk) == '[')))
137                     {
138                         stk  =  pop(stk, val)  ;
139                     }
140                     else
141                     {
142                         while(!empty_stack(stk))
143                             stk  =  pop(stk, val)  ;  // drain the stack
144
145                         return  false  ;
146                     }
147                 }
148             }
149         }
150
151         i++  ;
152     }
```

```
153
154     if(empty_stack(stk))
155         return  true   ;
156     else
157     {
158         while(!empty_stack(stk))
159             stk  =  pop(stk, val)  ;// drain the stack
160
161         return  false          ;
162     }
163 }
164
165 int  main()
166 {
167     char  str[50]   ;
168
169     cin  >>  str  ;
170
171     if(parentheses_match(str))
172     {
173         cout  <<  "The expression matches parenthesis."        ;
174     }
175     else
176     {
177         cout  <<  "The expression doesn't match parenthesis."  ;
178     }
179
180     return  0  ;
181 }
```