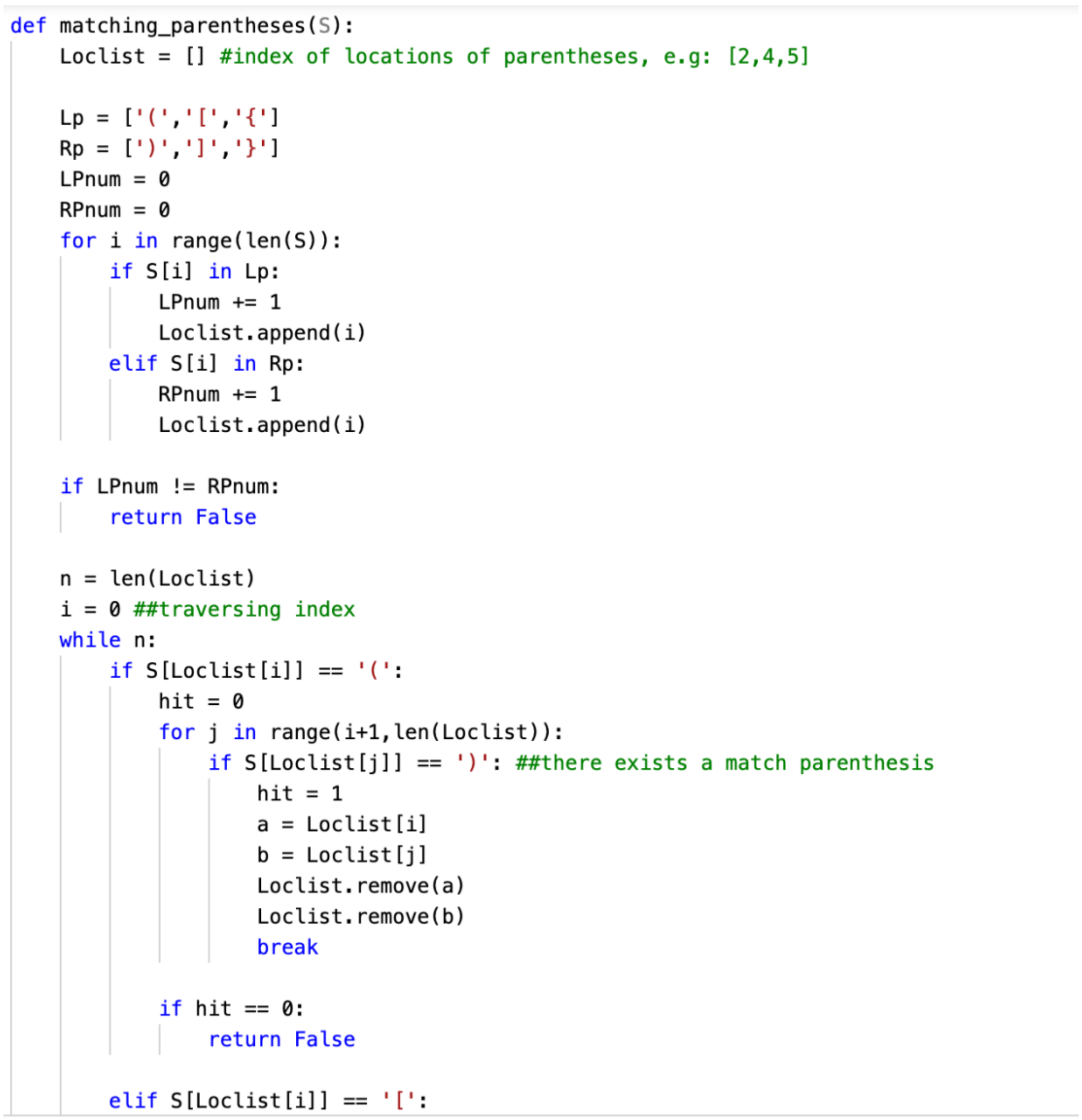
Problem 1

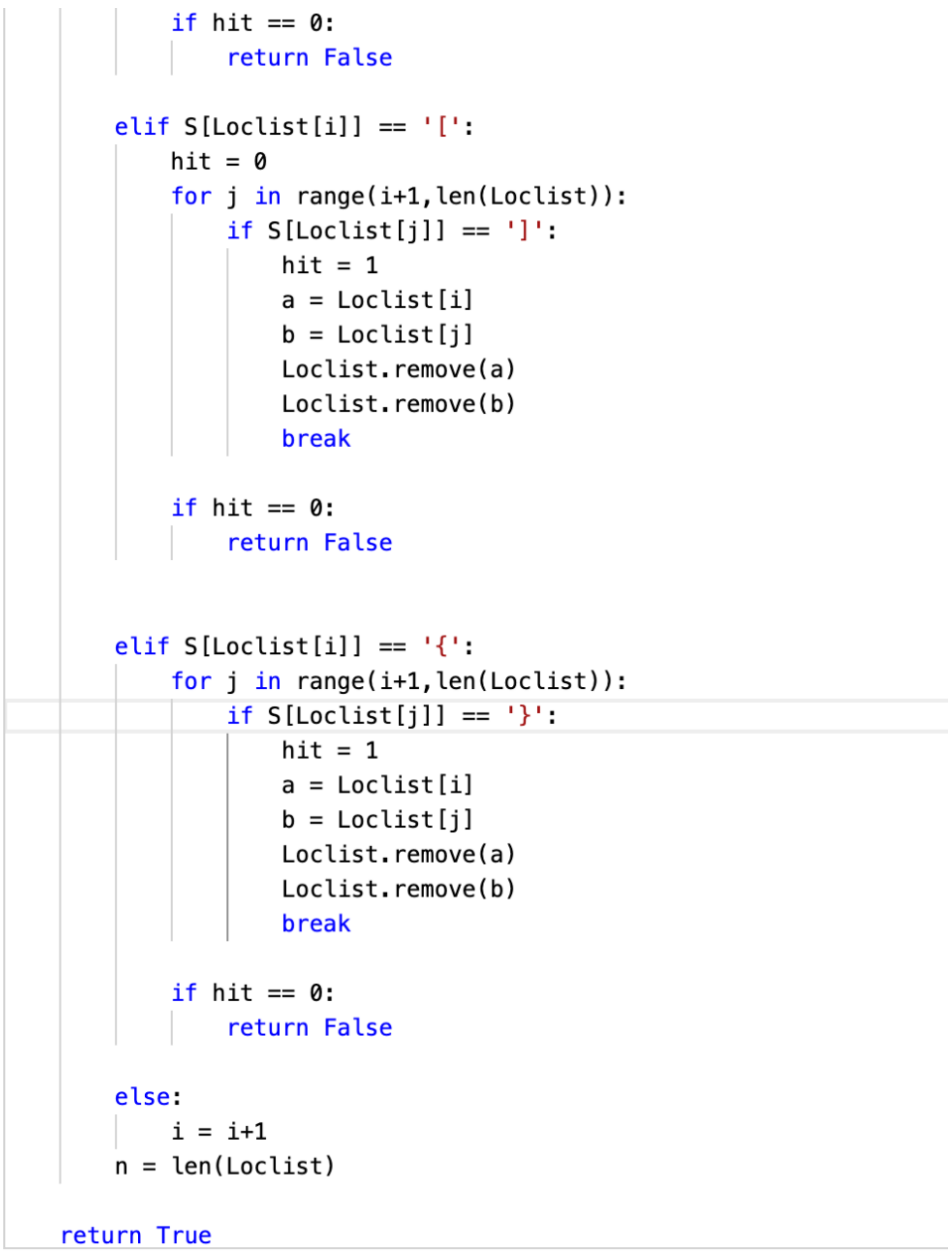
Step 1

1. Checking if the number of right parentheses is equal to the number of right parentheses
2. Storing indexes of parentheses into a list.
3. Traversing through that list. Set i = 0. While the size of the list is not zero, if list[i] is a left parenthesis, then checking the rest of the list to see if there exists a corresponding right parenthesis. Else “false” will be returned. Else if list[i] is a right parenthesis, i++. Every iteration will update the size of the list so it will not get into an infinite loop.
4. Return true.

Step 2 (Python version)



```python
def matching_parentheses(S):
    Loclist = [] #index of locations of parentheses, e.g: [2,4,5]

    Lp = ['(','[','{']
    Rp = [')',']','}']
    LPnum = 0
    RPnum = 0
    for i in range(len(S)):
        if S[i] in Lp:
            LPnum += 1
            Loclist.append(i)
        elif S[i] in Rp:
            RPnum += 1
            Loclist.append(i)

    if LPnum != RPnum:
        return False

    n = len(Loclist)
    i = 0 ##traversing index
    while n:
        if S[Loclist[i]] == '(':
            hit = 0
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == ')': ##there exists a match parenthesis
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False

        elif S[Loclist[i]] == '[':
```

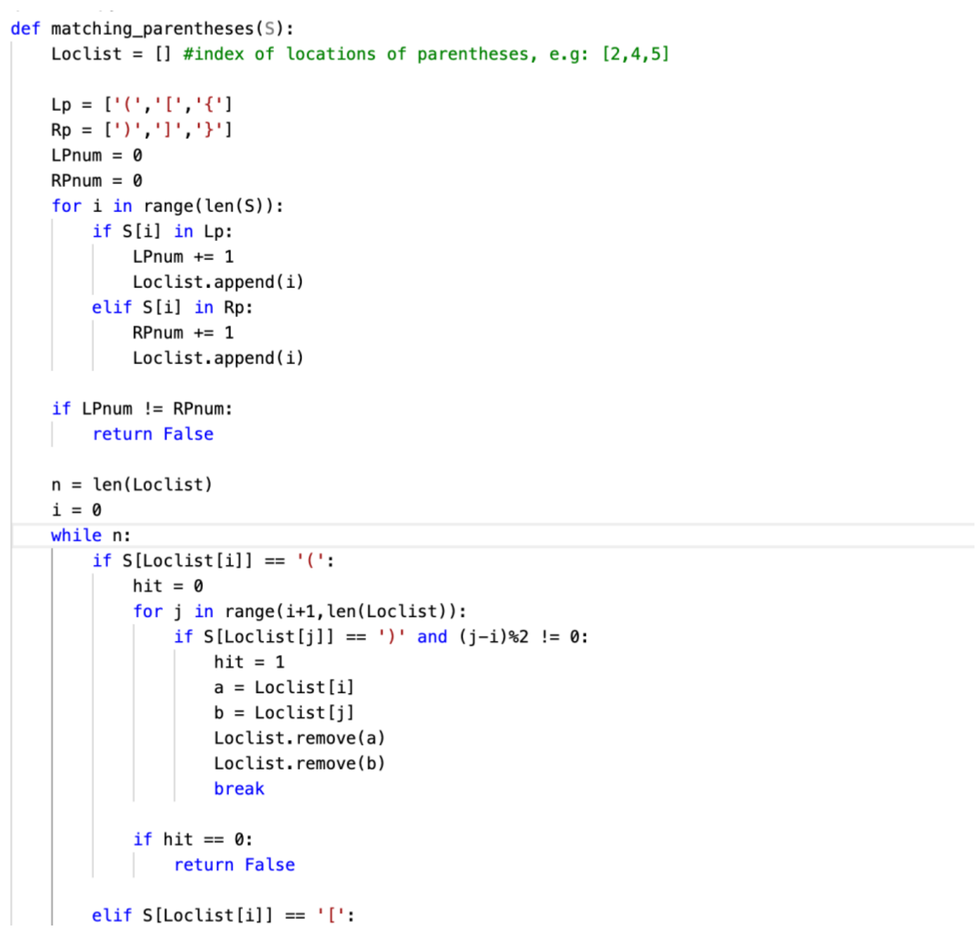


```
            if hit == 0:
                return False

        elif S[Loclist[i]] == '[':
            hit = 0
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == ']':
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False


        elif S[Loclist[i]] == '{':
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == '}':
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False

        else:
            i = i+1
        n = len(Loclist)

    return True
```

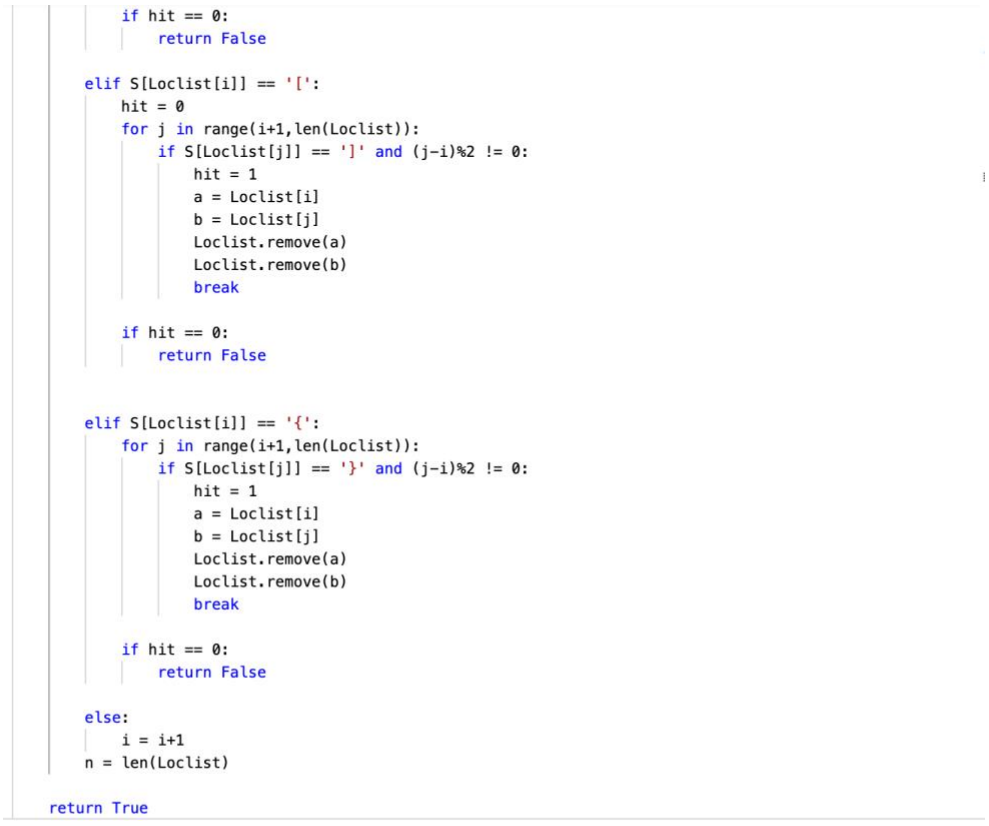
Step 3 (Python version)
Same as step 2 since I didn't encounter any syntax errors.

Step 4 (Python version)
Missing the condition check that if the order of parentheses is correct. Before this process, my program may pass the case "ab(ef[cd)]", which is actually false. Because I forget to check if the number of parentheses between two matching parentheses is also an even number. So, I added a condition after every "if S[Loclist[i]] == a right parenthesis"

Forgetting to check the situation of an empty string.



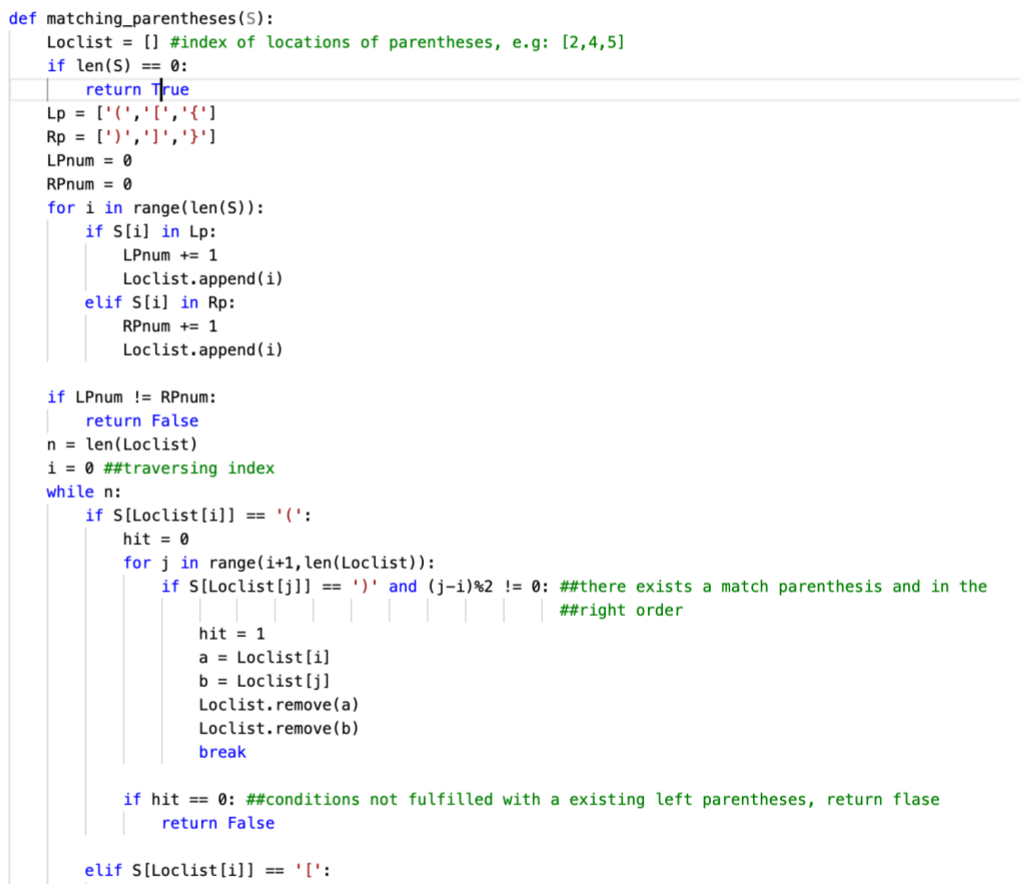
```python
def matching_parentheses(S):
    Loclist = [] #index of locations of parentheses, e.g: [2,4,5]

    Lp = ['(','[','{']
    Rp = [')',']','}']
    LPnum = 0
    RPnum = 0
    for i in range(len(S)):
        if S[i] in Lp:
            LPnum += 1
            Loclist.append(i)
        elif S[i] in Rp:
            RPnum += 1
            Loclist.append(i)

    if LPnum != RPnum:
        return False

    n = len(Loclist)
    i = 0
    while n:
        if S[Loclist[i]] == '(':
            hit = 0
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == ')' and (j-i)%2 != 0:
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False

        elif S[Loclist[i]] == '[':
```



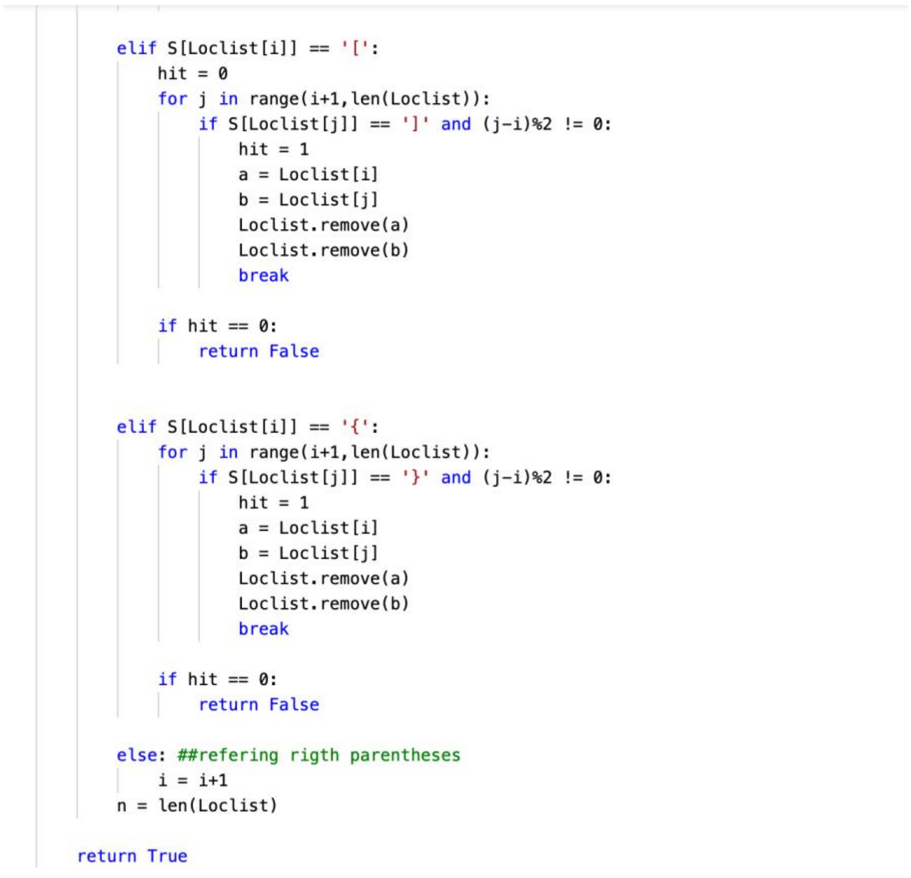
```python
            if hit == 0:
                return False
```

```python
        elif S[Loclist[i]] == '[':
            hit = 0
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == ']' and (j-i)%2 != 0:
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False


        elif S[Loclist[i]] == '{':
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == '}' and (j-i)%2 != 0:
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False

        else:
            i = i+1
        n = len(Loclist)

    return True
```

Step 5 (Python version)

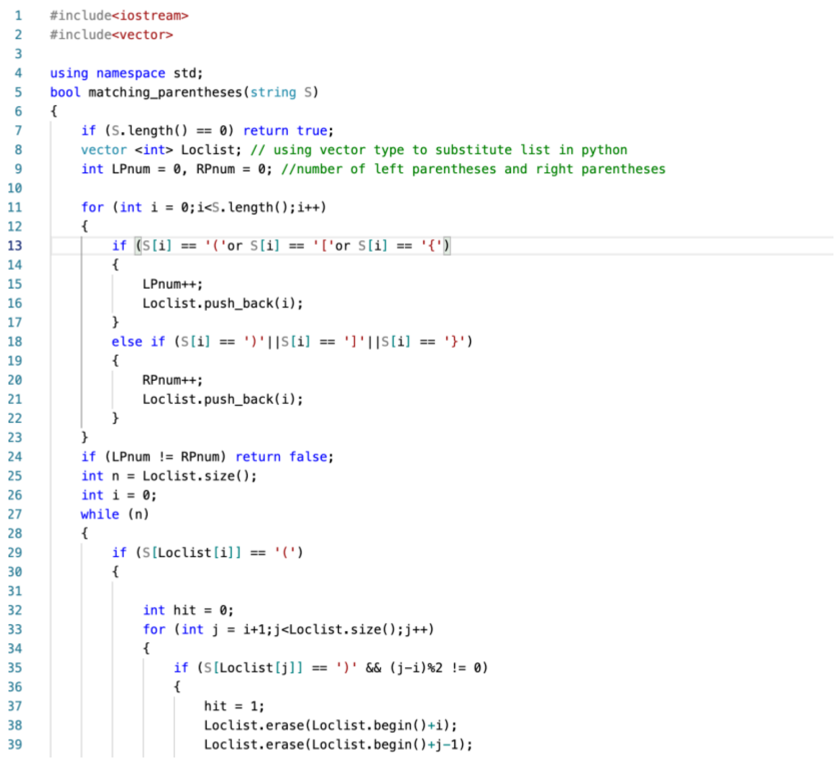


```python
def matching_parentheses(S):
    Loclist = [] #index of locations of parentheses, e.g: [2,4,5]
    if len(S) == 0:
        return True
    Lp = ['(','[','{']
    Rp = [')',']','}']
    LPnum = 0
    RPnum = 0
    for i in range(len(S)):
        if S[i] in Lp:
            LPnum += 1
            Loclist.append(i)
        elif S[i] in Rp:
            RPnum += 1
            Loclist.append(i)

    if LPnum != RPnum:
        return False
    n = len(Loclist)
    i = 0 ##traversing index
    while n:
        if S[Loclist[i]] == '(':
            hit = 0
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == ')' and (j-i)%2 != 0: ##there exists a match parenthesis and in the
                                                          ##right order
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0: ##conditions not fulfilled with a existing left parentheses, return flase
                return False

        elif S[Loclist[i]] == '[':
```

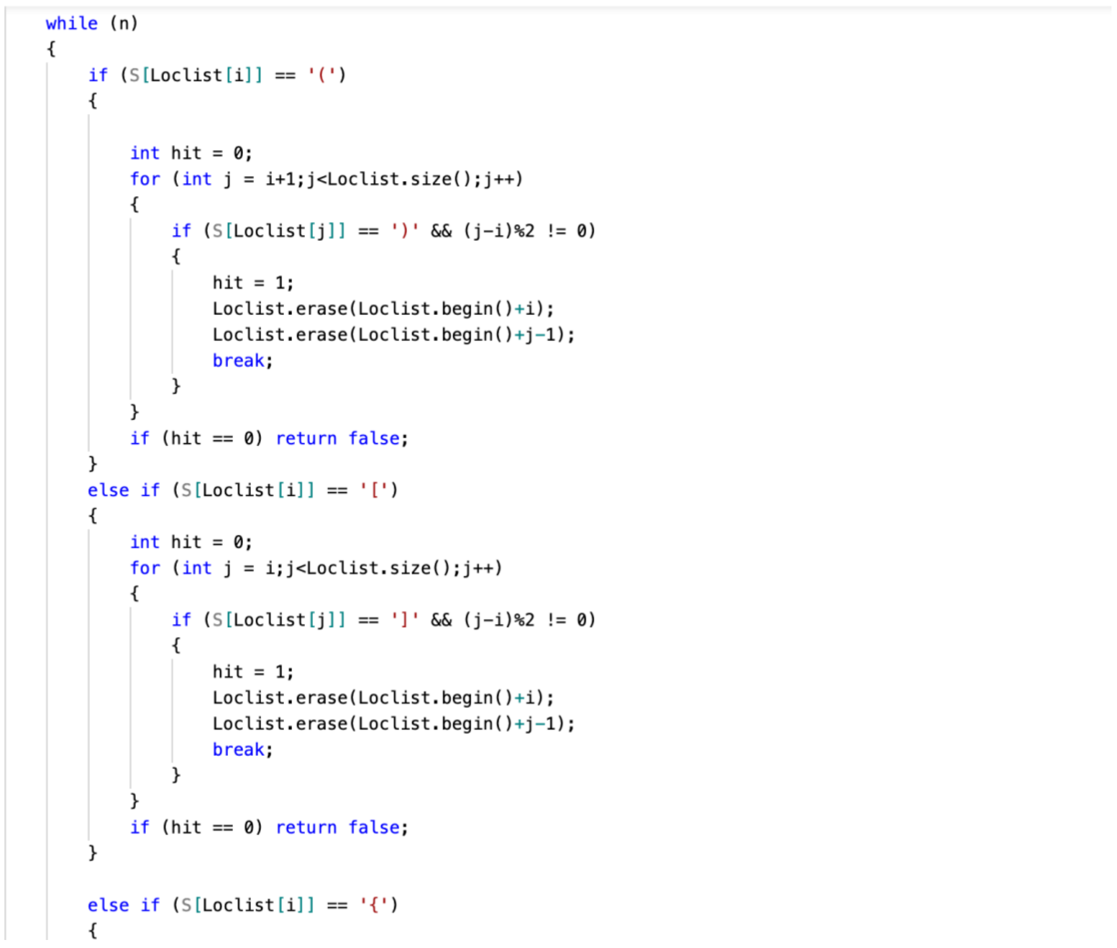


```python
        elif S[Loclist[i]] == '[':
            hit = 0
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == ']' and (j-i)%2 != 0:
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False


        elif S[Loclist[i]] == '{':
            for j in range(i+1,len(Loclist)):
                if S[Loclist[j]] == '}' and (j-i)%2 != 0:
                    hit = 1
                    a = Loclist[i]
                    b = Loclist[j]
                    Loclist.remove(a)
                    Loclist.remove(b)
                    break

            if hit == 0:
                return False

        else: ##refering rigth parentheses
            i = i+1
        n = len(Loclist)

    return True
```

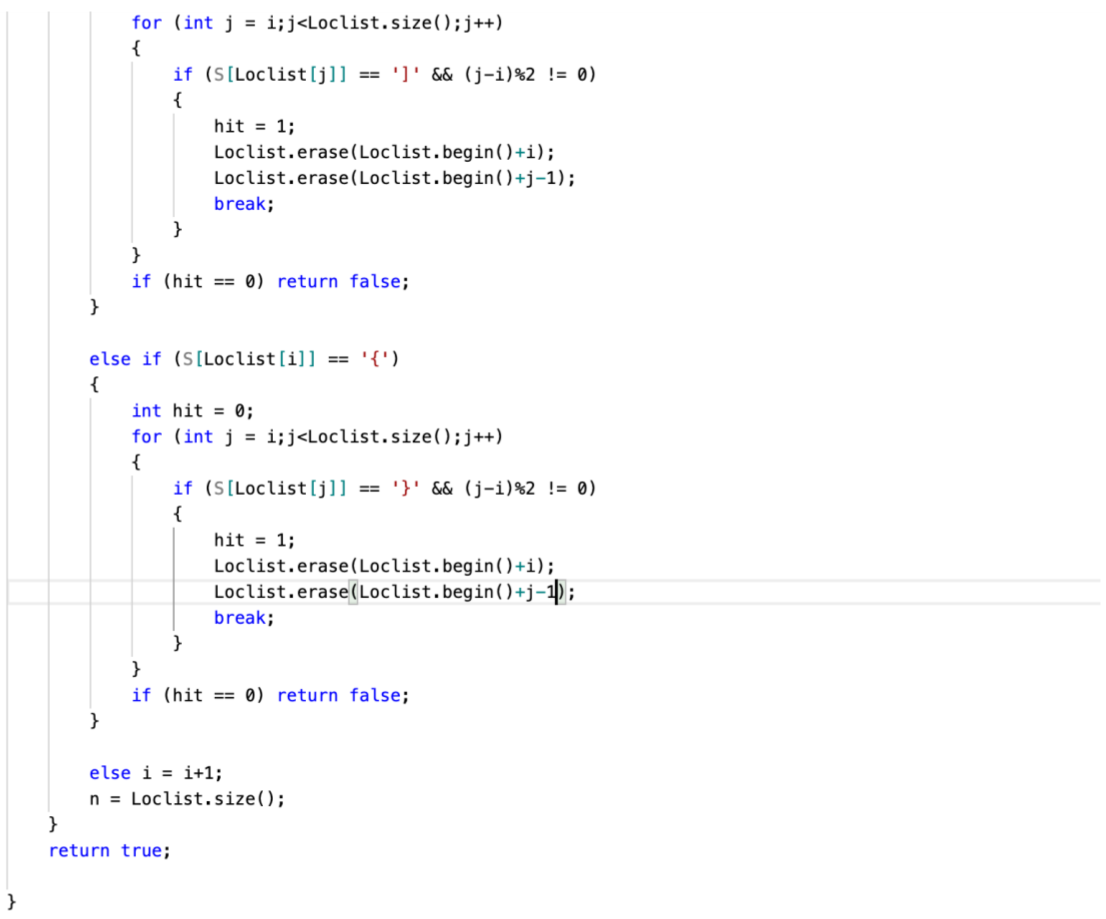
Step 2 (C++ version)



```cpp
#include<iostream>
#include<vector>

using namespace std;
bool matching_parentheses(string S)
{
    if (S.length() == 0) return true;
    vector <int> Loclist; // using vector type to substitute list in python
    int LPnum = 0, RPnum = 0; //number of left parentheses and right parentheses

    for (int i = 0;i<S.length();i++)
    {
        if (S[i] == '('or S[i] == '['or S[i] == '{')
        {
            LPnum++;
            Loclist.push_back(i);
        }
        else if (S[i] == ')'||S[i] == ']'||S[i] == '}')
        {
            RPnum++;
            Loclist.push_back(i);
        }
    }
    if (LPnum != RPnum) return false;
    int n = Loclist.size();
    int i = 0;
    while (n)
    {
        if (S[Loclist[i]] == '(')
        {

            int hit = 0;
            for (int j = i+1;j<Loclist.size();j++)
            {
                if (S[Loclist[j]] == ')' && (j-i)%2 != 0)
                {
                    hit = 1;
                    Loclist.erase(Loclist.begin()+i);
                    Loclist.erase(Loclist.begin()+j-1);
```



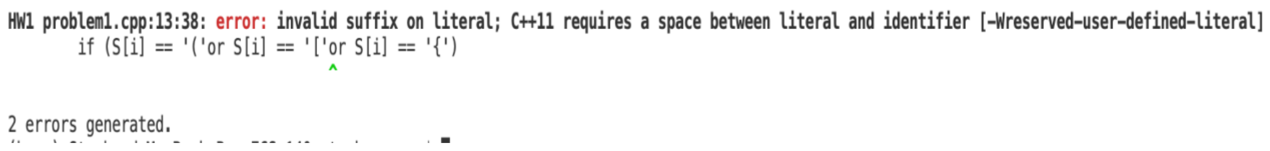
```cpp
while (n)
{
    if (S[Loclist[i]] == '(')
    {

        int hit = 0;
        for (int j = i+1;j<Loclist.size();j++)
        {
            if (S[Loclist[j]] == ')' && (j-i)%2 != 0)
            {
                hit = 1;
                Loclist.erase(Loclist.begin()+i);
                Loclist.erase(Loclist.begin()+j-1);
                break;
            }
        }
        if (hit == 0) return false;
    }
    else if (S[Loclist[i]] == '[')
    {
        int hit = 0;
        for (int j = i;j<Loclist.size();j++)
        {
            if (S[Loclist[j]] == ']' && (j-i)%2 != 0)
            {
                hit = 1;
                Loclist.erase(Loclist.begin()+i);
                Loclist.erase(Loclist.begin()+j-1);
                break;
            }
        }
        if (hit == 0) return false;
    }

    else if (S[Loclist[i]] == '{')
    {
```



```
            for (int j = i;j<Loclist.size();j++)
            {
                if (S[Loclist[j]] == ']' && (j-i)%2 != 0)
                {
                    hit = 1;
                    Loclist.erase(Loclist.begin()+i);
                    Loclist.erase(Loclist.begin()+j-1);
                    break;
                }
            }
            if (hit == 0) return false;
        }

        else if (S[Loclist[i]] == '{')
        {
            int hit = 0;
            for (int j = i;j<Loclist.size();j++)
            {
                if (S[Loclist[j]] == '}' && (j-i)%2 != 0)
                {
                    hit = 1;
                    Loclist.erase(Loclist.begin()+i);
                    Loclist.erase(Loclist.begin()+j-1);
                    break;
                }
            }
            if (hit == 0) return false;
        }

        else i = i+1;
        n = Loclist.size();
    }
    return true;

}
```

Error:

At first, I used “and” “or” instead of “&&” “||”. Them the syntax error popped out.



```
HW1 problem1.cpp:13:38: error: invalid suffix on literal; C++11 requires a space between literal and identifier [-Wreserved-user-defined-literal]
        if (S[i] == '('or S[i] == '['or S[i] == '{')
                                     ^

2 errors generated.
```

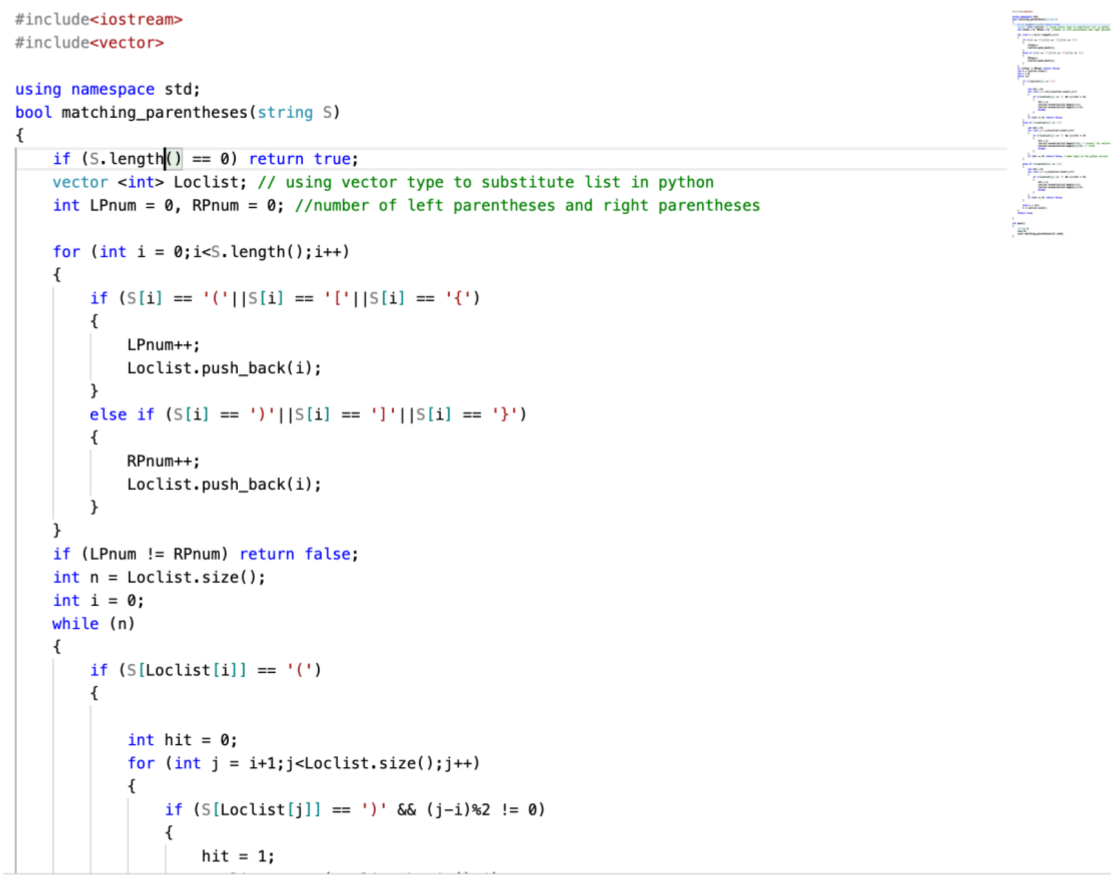
Step 3 (C++ version)

Same as step except “or”s are changed to “||”s

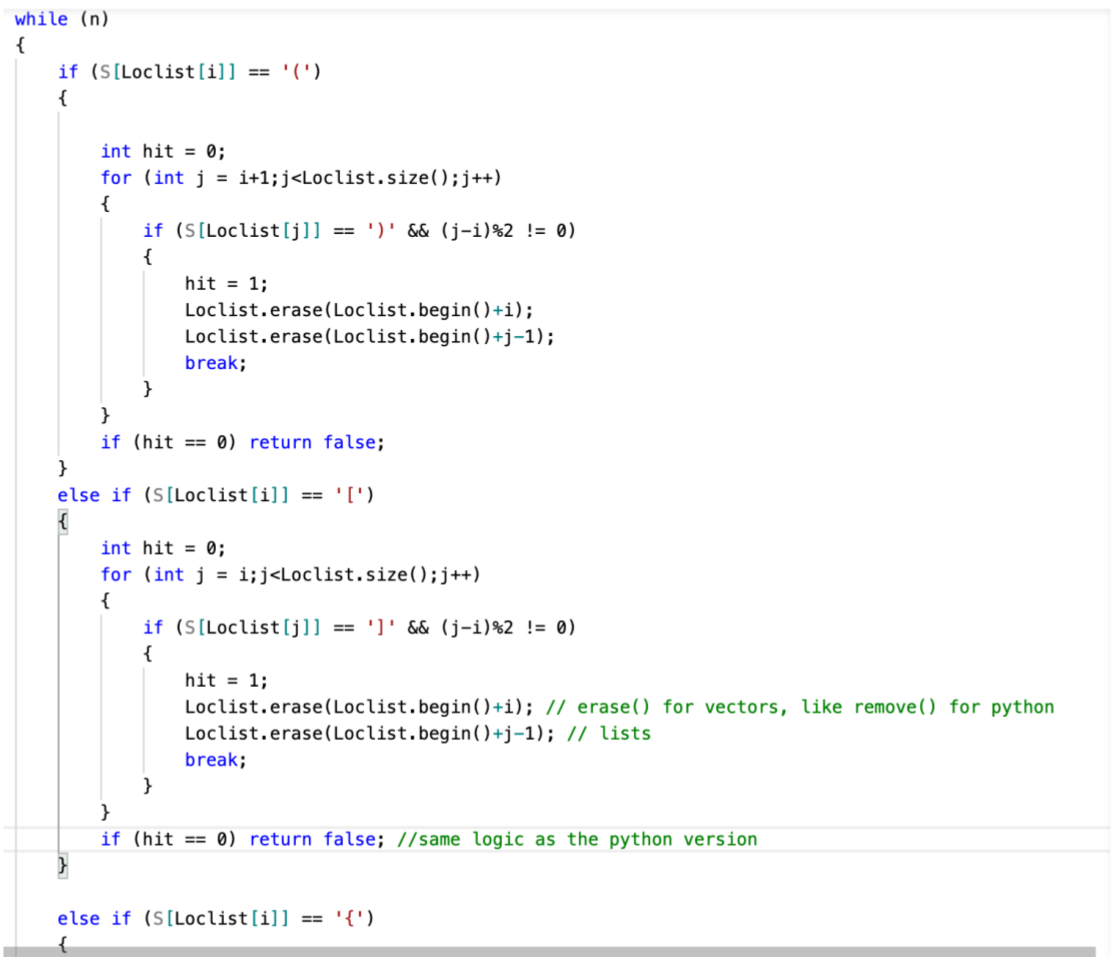
Step 4 (C++ version)

Since the logic has been already fixed when I was doing the python version, this c++ version passed all the test cases.

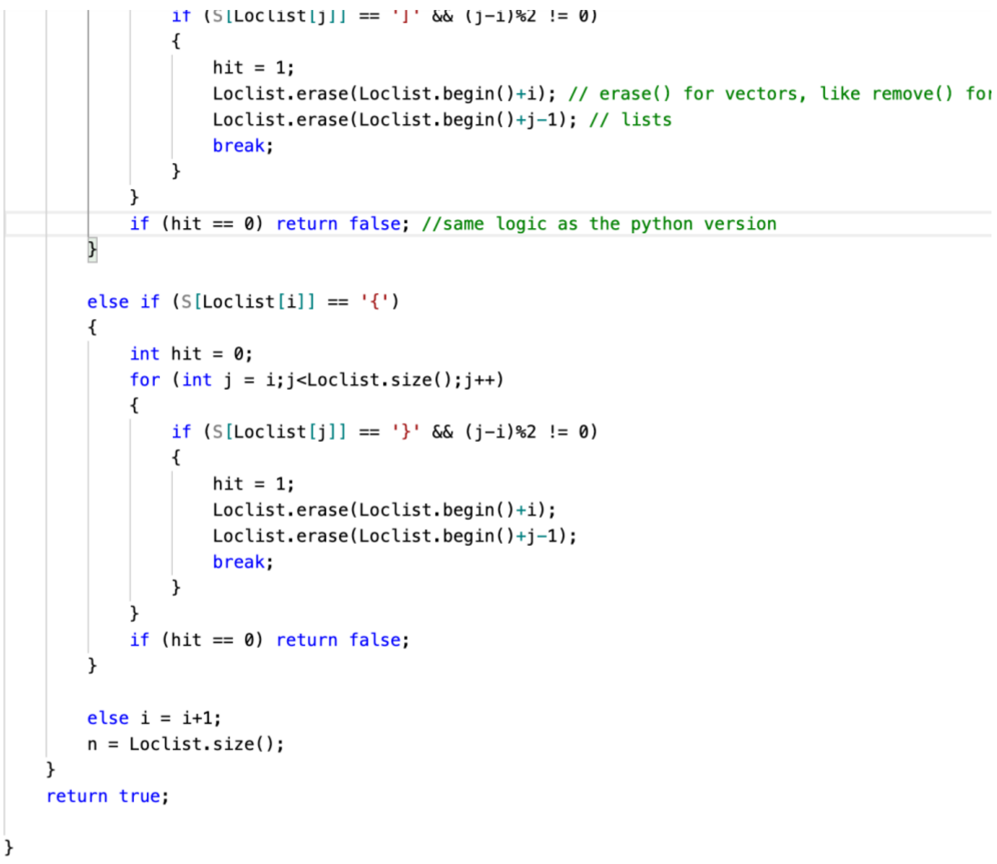
Step 5 (C++ version)



```cpp
#include<iostream>
#include<vector>

using namespace std;
bool matching_parentheses(string S)
{
    if (S.length() == 0) return true;
    vector <int> Loclist; // using vector type to substitute list in python
    int LPnum = 0, RPnum = 0; //number of left parentheses and right parentheses

    for (int i = 0;i<S.length();i++)
    {
        if (S[i] == '('||S[i] == '['||S[i] == '{')
        {
            LPnum++;
            Loclist.push_back(i);
        }
        else if (S[i] == ')'||S[i] == ']'||S[i] == '}')
        {
            RPnum++;
            Loclist.push_back(i);
        }
    }
    if (LPnum != RPnum) return false;
    int n = Loclist.size();
    int i = 0;
    while (n)
    {
        if (S[Loclist[i]] == '(')
        {

            int hit = 0;
            for (int j = i+1;j<Loclist.size();j++)
            {
                if (S[Loclist[j]] == ')' && (j-i)%2 != 0)
                {
                    hit = 1;
```

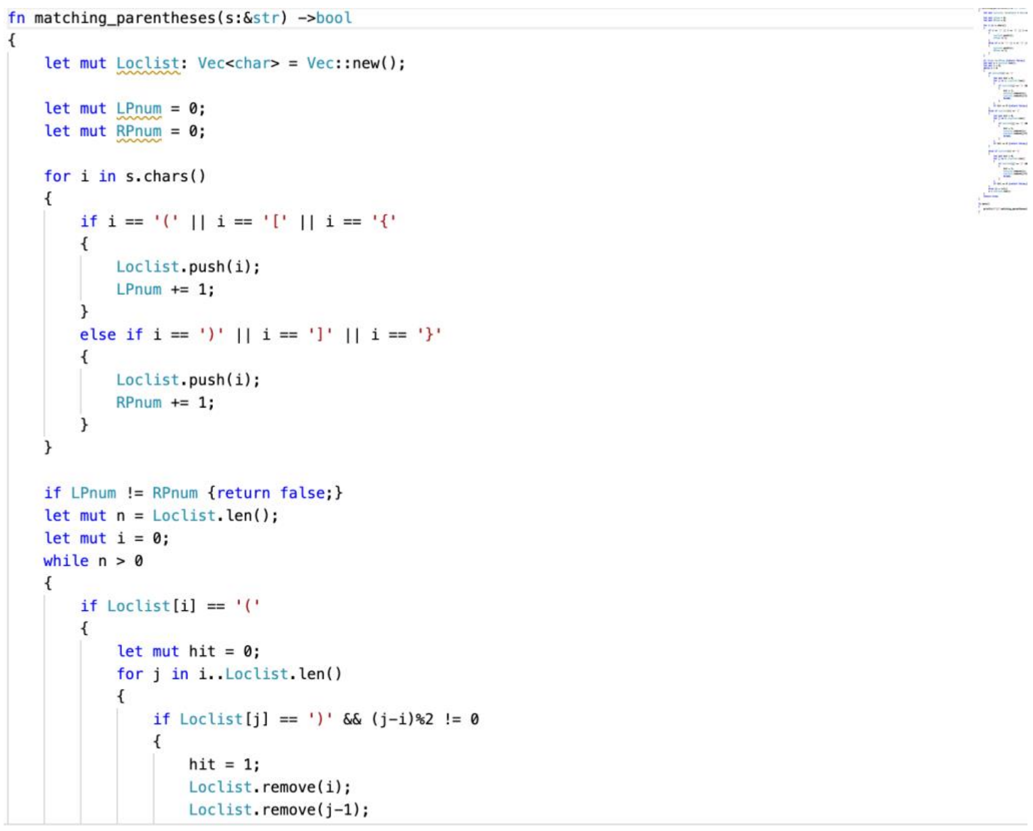


```cpp
while (n)
{
    if (S[Loclist[i]] == '(')
    {

        int hit = 0;
        for (int j = i+1;j<Loclist.size();j++)
        {
            if (S[Loclist[j]] == ')' && (j-i)%2 != 0)
            {
                hit = 1;
                Loclist.erase(Loclist.begin()+i);
                Loclist.erase(Loclist.begin()+j-1);
                break;
            }
        }
        if (hit == 0) return false;
    }
    else if (S[Loclist[i]] == '[')
    {
        int hit = 0;
        for (int j = i;j<Loclist.size();j++)
        {
            if (S[Loclist[j]] == ']' && (j-i)%2 != 0)
            {
                hit = 1;
                Loclist.erase(Loclist.begin()+i); // erase() for vectors, like remove() for python
                Loclist.erase(Loclist.begin()+j-1); // lists
                break;
            }
        }
        if (hit == 0) return false; //same logic as the python version
    }

    else if (S[Loclist[i]] == '{')
    {
```

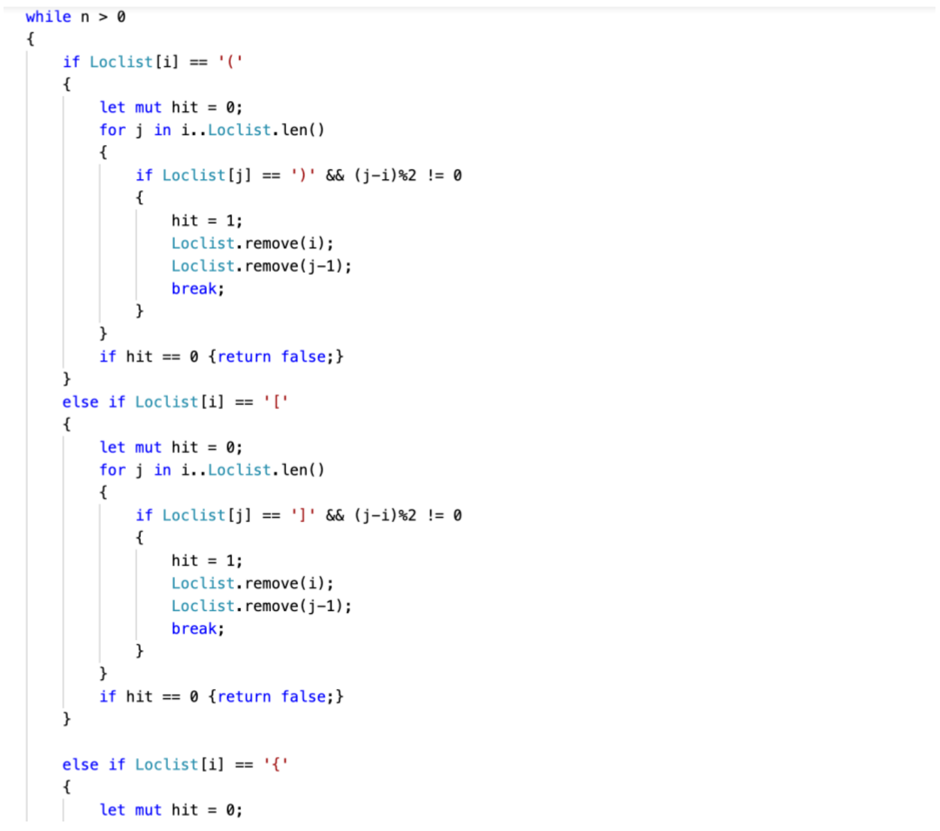


```cpp
                if (S[Loclist[j]] == ']' && (j-i)%2 != 0)
                {
                    hit = 1;
                    Loclist.erase(Loclist.begin()+i); // erase() for vectors, like remove() fo
                    Loclist.erase(Loclist.begin()+j-1); // lists
                    break;
                }
            }
            if (hit == 0) return false; //same logic as the python version
        }

        else if (S[Loclist[i]] == '{')
        {
            int hit = 0;
            for (int j = i;j<Loclist.size();j++)
            {
                if (S[Loclist[j]] == '}' && (j-i)%2 != 0)
                {
                    hit = 1;
                    Loclist.erase(Loclist.begin()+i);
                    Loclist.erase(Loclist.begin()+j-1);
                    break;
                }
            }
            if (hit == 0) return false;
        }

        else i = i+1;
        n = Loclist.size();
    }
    return true;

}
```

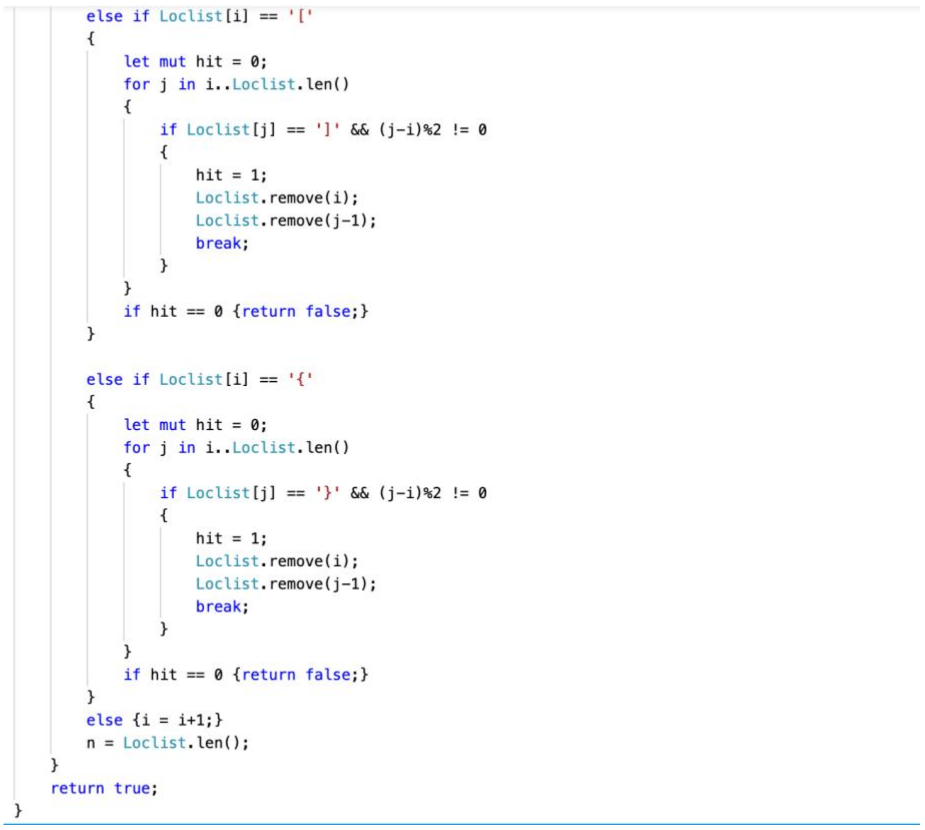
Step 2 (Rust version)



```rust
fn matching_parentheses(s:&str) ->bool
{
    let mut Loclist: Vec<char> = Vec::new();

    let mut LPnum = 0;
    let mut RPnum = 0;

    for i in s.chars()
    {
        if i == '(' || i == '[' || i == '{'
        {
            Loclist.push(i);
            LPnum += 1;
        }
        else if i == ')' || i == ']' || i == '}'
        {
            Loclist.push(i);
            RPnum += 1;
        }
    }

    if LPnum != RPnum {return false;}
    let mut n = Loclist.len();
    let mut i = 0;
    while n > 0
    {
        if Loclist[i] == '('
        {
            let mut hit = 0;
            for j in i..Loclist.len()
            {
                if Loclist[j] == ')' && (j-i)%2 != 0
                {
                    hit = 1;
                    Loclist.remove(i);
                    Loclist.remove(j-1);
```



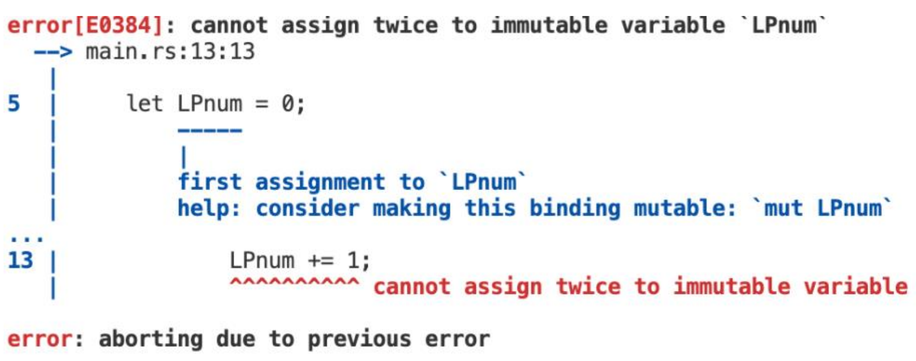
```
while n > 0
{
    if Loclist[i] == '('
    {
        let mut hit = 0;
        for j in i..Loclist.len()
        {
            if Loclist[j] == ')' && (j-i)%2 != 0
            {
                hit = 1;
                Loclist.remove(i);
                Loclist.remove(j-1);
                break;
            }
        }
        if hit == 0 {return false;}
    }
    else if Loclist[i] == '['
    {
        let mut hit = 0;
        for j in i..Loclist.len()
        {
            if Loclist[j] == ']' && (j-i)%2 != 0
            {
                hit = 1;
                Loclist.remove(i);
                Loclist.remove(j-1);
                break;
            }
        }
        if hit == 0 {return false;}
    }

    else if Loclist[i] == '{'
    {
        let mut hit = 0;
```



```
        else if Loclist[i] == '['
        {
            let mut hit = 0;
            for j in i..Loclist.len()
            {
                if Loclist[j] == ']' && (j-i)%2 != 0
                {
                    hit = 1;
                    Loclist.remove(i);
                    Loclist.remove(j-1);
                    break;
                }
            }
            if hit == 0 {return false;}
        }

        else if Loclist[i] == '{'
        {
            let mut hit = 0;
            for j in i..Loclist.len()
            {
                if Loclist[j] == '}' && (j-i)%2 != 0
                {
                    hit = 1;
                    Loclist.remove(i);
                    Loclist.remove(j-1);
                    break;
                }
            }
            if hit == 0 {return false;}
        }
        else {i = i+1;}
        n = Loclist.len();
    }
    return true;
}
```

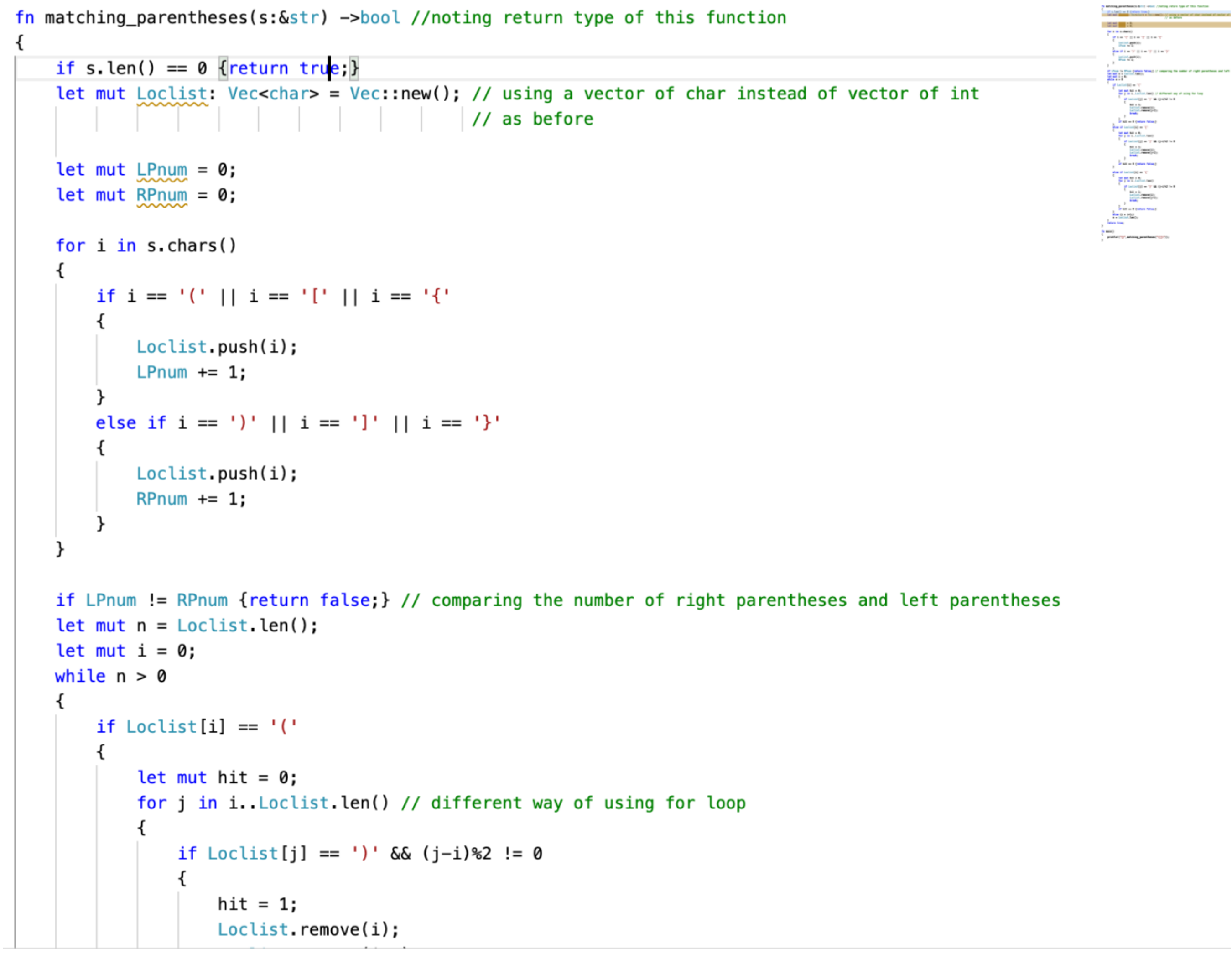
Step 3 (Rust version)

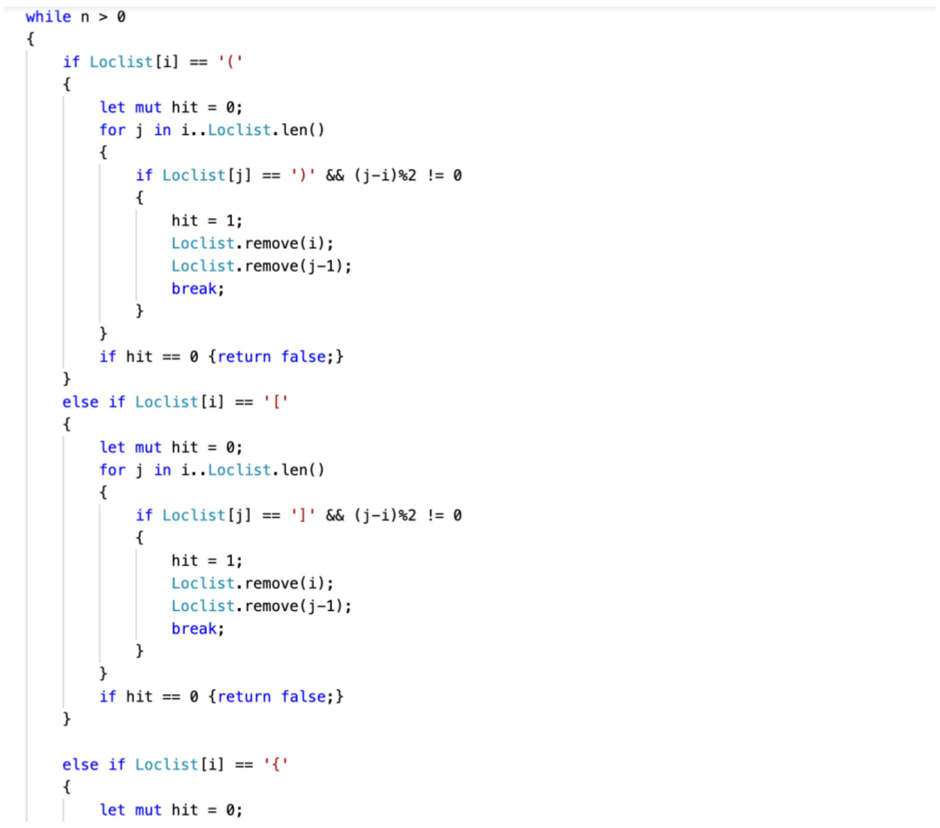
Since this is my first-time writing Rust code, forgetting to add “mut” in front of variables whose values are changed soon is the most frequent error I encounter.


```
error[E0384]: cannot assign twice to immutable variable `LPnum`
  --> main.rs:13:13
   |
5  |     let LPnum = 0;
   |         -----
   |         |
   |         first assignment to `LPnum`
   |         help: consider making this binding mutable: `mut LPnum`
...
13 |             LPnum += 1;
   |             ^^^^^^^^^^ cannot assign twice to immutable variable

error: aborting due to previous error
```

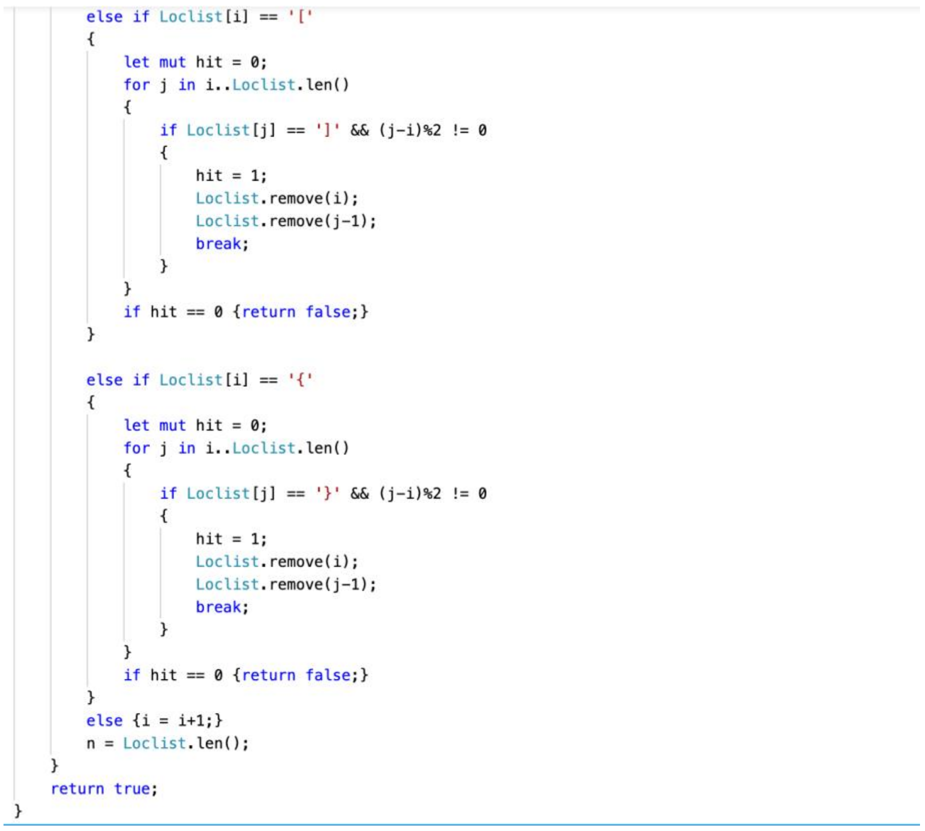
Step 4 (Rust version)
Forgetting to check the situation of an empty input string

Step 5 (Rust version)


```
fn matching_parentheses(s:&str) ->bool //noting return type of this function
{
    if s.len() == 0 {return true;}
    let mut Loclist: Vec<char> = Vec::new(); // using a vector of char instead of vector of int
                                             // as before

    let mut LPnum = 0;
    let mut RPnum = 0;

    for i in s.chars()
    {
        if i == '(' || i == '[' || i == '{'
        {
            Loclist.push(i);
            LPnum += 1;
        }
        else if i == ')' || i == ']' || i == '}'
        {
            Loclist.push(i);
            RPnum += 1;
        }
    }

    if LPnum != RPnum {return false;} // comparing the number of right parentheses and left parentheses
    let mut n = Loclist.len();
    let mut i = 0;
    while n > 0
    {
        if Loclist[i] == '('
        {
            let mut hit = 0;
            for j in i..Loclist.len() // different way of using for loop
            {
                if Loclist[j] == ')' && (j-i)%2 != 0
                {
                    hit = 1;
                    Loclist.remove(i);
```



```
while n > 0
{
    if Loclist[i] == '('
    {
        let mut hit = 0;
        for j in i..Loclist.len()
        {
            if Loclist[j] == ')' && (j-i)%2 != 0
            {
                hit = 1;
                Loclist.remove(i);
                Loclist.remove(j-1);
                break;
            }
        }
        if hit == 0 {return false;}
    }
    else if Loclist[i] == '['
    {
        let mut hit = 0;
        for j in i..Loclist.len()
        {
            if Loclist[j] == ']' && (j-i)%2 != 0
            {
                hit = 1;
                Loclist.remove(i);
                Loclist.remove(j-1);
                break;
            }
        }
        if hit == 0 {return false;}
    }

    else if Loclist[i] == '{'
    {
        let mut hit = 0;
```



```
        else if Loclist[i] == '['
        {
            let mut hit = 0;
            for j in i..Loclist.len()
            {
                if Loclist[j] == ']' && (j-i)%2 != 0
                {
                    hit = 1;
                    Loclist.remove(i);
                    Loclist.remove(j-1);
                    break;
                }
            }
            if hit == 0 {return false;}
        }

        else if Loclist[i] == '{'
        {
            let mut hit = 0;
            for j in i..Loclist.len()
            {
                if Loclist[j] == '}' && (j-i)%2 != 0
                {
                    hit = 1;
                    Loclist.remove(i);
                    Loclist.remove(j-1);
                    break;
                }
            }
            if hit == 0 {return false;}
        }
        else {i = i+1;}
        n = Loclist.len();
    }
    return true;
}
```

Step 6 (for all 3 versions)

These are all of the test cases except the given test cases in the prompt I used to test the three version of the program:

1. (abcd)
2. (ab(cd)fg)g
3. asd(asf[adsf)daf]
4. as]asdas)das}]asd
5. (as(d)]asd)
6. gdfd((g(g(
7. ()[gsdg]{}
8. 3(sdaf[g{g}]h)h
9. as{f[])as