```
#include <bits/stdc++.h>
using namespace std;

int main() {

        string  city = "yash";

        stack<char>st;

        for(int i=0;i<city.size();i++)
        {
               st.push(city[i]);
        }

        string ans="";

        while(st.size()>0)
        {
              ans+= st.top();
              st.pop();
        }
        cout<<ans<<endl;


            return 0;
}
```

```
/*package whatever //do not write package name here */

import java.util.*;

class GFG {
	public static void main (String[] args)
	{
		Stack<Character> st= new Stack();

		String city= "mumbai";

		for(int i=0;i<city.length();i++)
		{
		    st.push( city.charAt(i));
		}

		String ans="";
		while(st.size()>0)
		{
		    ans+= st.peek();
		    st.pop();
		}

		System.out.println(ans);
	}
}
```

```
/*package whatever //do not write package name here */

import java.util.*;

class GFG {
	public static void main (String[] args)
	{
		StringBuilder sb= new StringBuilder("ya");

		System.out.println(  sb );

		System.out.println(   sb.hashCode()   );



		sb.append("s");

		System.out.println(  sb);

		System.out.println( sb.hashCode()  );
	}
}
```

Remove All Adjacent Duplicates In String - LeetCode

Java Code :

```
class Solution {
    public String removeDuplicates(String str) {
        Stack<Character> st = new Stack<>();

        for (int i = 0; i < str.length(); i++) {
            if (st.isEmpty()) {
                st.push(str.charAt(i));
            } else {
                if (st.peek() == str.charAt(i)) {
                    st.pop();
                } else {
                    st.push(str.charAt(i));
                }
            }
        }

        StringBuilder ans = new StringBuilder();

        while (!st.isEmpty()) {
            ans.append(st.pop());
        }

        return ans.reverse().toString();
    }
}
```

```
class Solution:
    def removeDuplicates(self, s):
        st = []

        for char in s:
            if not st:
                st.append(char)
            else:
                if st[-1] == char:
                    st.pop()
                else:
                    st.append(char)

        ans = ''.join(reversed(st))
        return ans
```

[Reverse Words in a String - LeetCode]

C++ :

```
class Solution {
public:
    string reverseWords(string text) {

        stack<string>st;

        string temp="";

        for(int i=0;i<text.size();i++)
        {
              char ch= text[i];

             if(ch!=' ')
             {
                 // ch is b/w  a to z
                   temp+= ch;
             }
            else
            {
                // char is a space
                  if(temp.size()>0)
                  {
                       st.push(temp);
                       temp="";
                  }
            }

        }

        if(temp.size()>0)
        {
              st.push(temp);
            temp="";
        }

        string ans="";

        while(st.size()>0)
        {
            string word= st.top();
            st.pop();

            if(st.size()>0)
            {
               ans+= word+ " ";
            }
          else
          {
               ans+= word;
          }
        }

        return ans;


    }
};
```

Java :

```
class Solution {

      public String reverseWords(String text) {
    Stack<String> st = new Stack<>();
    StringBuilder temp = new StringBuilder();

    for (int i = 0; i < text.length(); i++) {
        char ch = text.charAt(i);

        if (ch != ' ') {
            // ch is between 'a' to 'z'
            temp.append(ch);
        } else {
            // char is a space
            if (temp.length() > 0) {
                st.push(temp.toString());
                temp.setLength(0);
            }
        }
    }

    if (temp.length() > 0) {
        st.push(temp.toString());
        temp.setLength(0);
    }

    StringBuilder ans = new StringBuilder();

    while (!st.isEmpty()) {
        String word = st.pop();

        if (!st.isEmpty()) {
            ans.append(word).append(" ");
        } else {
            ans.append(word);
        }
    }

    return ans.toString();
  }
}
```

[Remove All Adjacent Duplicates in String II - LeetCode]

```
class Solution {
public:
    string removeDuplicates(string text, int k)
    {
           stack<pair<char,int>> st;

           for(int i=0;i< text.size(); i++)
           {

                 char ch= text[i];

                if(st.size()==0)
                {
                      st.push( {ch,1} );
                }
               else
               {
                      if(st.top().first == ch)
                      {
                            st.top().second++;

                              if(st.top().second==k)
                              {
                                    st.pop();
                              }
                      }
                       else
                       {
                            st.push( {ch,1} );
                       }
               }
           }

          string ans="";

          while(st.size()>0)
          {
                char ch= st.top().first;
                int freq= st.top().second;

                st.pop();

                for(int i=1;i<=freq;i++)
                {
                    ans+=ch;
                }
          }

          reverse(ans.begin(), ans.end());

          return ans;

    }
};
```

```
class Pair {
    public char first;
    public int second;

    public Pair(char first, int second)
    {
        this.first = first;
        this.second = second;
    }
}

public class Solution {
    public String removeDuplicates(String text, int k) {
        // Stack to store pairs of characters and their frequencies
        Stack<Pair> st = new Stack<>();

        // Iterate through each character in the input string
        for (int i = 0; i < text.length(); i++) {
            char ch = text.charAt(i);

            // If the stack is empty, push the current character with frequency 1
            if (st.isEmpty())
            {
                st.push(new Pair(ch, 1));
            } else
            {
                // If the current character is the same as the character at the top of the stack
                if (st.peek().first == ch)
                {
                    // Increment the frequency
                    st.peek().second++;

                    // If the frequency becomes equal to k, pop the element from the stack
                    if (st.peek().second == k)
                    {
                        st.pop();
                    }
                }
                else
                {
                    // If the current character is different, push it to the stack with frequency 1
                st.push(new Pair(ch, 1));
            }
        }
    }

    // Reconstruct the string from the characters left in the stack
    StringBuilder ans = new StringBuilder();
    while (st.isEmpty()==false)
    {
        char ch = st.peek().first;
        int freq = st.peek().second;

        st.pop();

        // Append the character to the result string based on its frequency
        for (int i = 1; i <= freq; i++)
        {
            ans.append(ch);
        }
    }

    // Reverse the result string and return
    return ans.reverse().toString();
  }
}
```