4. (17%) Write a function: **`string remove_a(string word)`** that removes all occurrences of the letter '**a**' from string **`word`**.

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

string remove_a(string word) {
  string new_word;
  for (int i = 0; i < word.length(); i++) {
    if (word[i] != 'a') {
      new_word = new_word + word[i];
    }
  }
  return new_word;









}
int main() {
   cout << remove_a("Isabella") << endl;
   return 0;
}
```

NAME: FIRST LAST

4. (15%) Write a function: **`void remove_e(string & sentence)`** that removes all occurrences of letter '**e**' from string **`sentence`** in place: in its original memory location in the caller function.

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

void remove_e(string & sentence);

int main()
{
    string sentence = "Hello hello";
    remove_e(sentence);
    cout << endl << sentence << endl;
    return 0;
}

void remove_e(string & s) {
    for (int i = 0; i < s.length(); i++)
    {
        if (s[i] == 'e') {
            s = s.substr(0, i) + s.substr(i + 1, s.length() - 1);
            i--; }
    }
}
```

This function receives a string argument, and splits it **into two strings** on the **first space** it finds. For example, "**Fortune favors the bold**" is split into "**Fortune**" and "**favors the bold**".

The two arguments passed by reference, **`before`** and **`after`**, will contain the two resulting halves of the string: before and after the space:

```
string line = "AAAA  BB CCC";
string beforeSpace;
string afterSpace;
splitOnSpace(line, beforeSpace, afterSpace);
```

After the function call, the second and the third argument variables have the following values:

```
beforeSpace == "AAAA" // contains everything before the first space
afterSpace  == " BB CCC" // contains everything after it

void splitOnSpace(string s, string & before, string & after) {
  // reset strings
    before = "";
    after = "";
  // accumulate before space
    int i = 0;
    while (i < s.size() && not isspace(s[i])) {
        before = before + s[i];
        i++;
    }
  // skip the space
    i++;
  // accumulate after space
    while (i < s.size()) {
        after = after + s[i];
        i++;
    }
}
```