Homework 5: Strings

CS16 - Winter 2021

Due:	Thursday, February 11, 2021 (11:59 PM PST)
Points:	100
Name:	`Tyler_Pruitt__________________________________________`
Homework buddy:	`______________________________________________________`

- You may collaborate on this homework with **at most** one person, an optional "homework buddy."
- **Submission instructions:** All questions are to be written (either by hand or typed) *in the provided spaces* and turned in as a single PDF on Gradescope. If you submit handwritten solutions write legibly. We reserve the right to give 0 points to answers we cannot read. When you submit your answer on Gradescope, **be sure to select which portions of your answer correspond to which problem** and clearly mark on the page itself which problem you are answering. We reserve the right to give 0 points to submissions that fail to do this.

1. (10 points) Which of these are correct usage (syntax) of a single statement on a string variable called `message`, and which of these are incorrect usage (and *very briefly* **why**). Variables `n` and `m` are `int` types.

a. (2 points) `message.erase(n, m);`

```
Yes, this is the correct usage of the erase string method because it just
modifies the string message and it has two variables integer variables
(start position and length) which here are correctly inputted as n and m.
```

b. (2 points) `message = message.erase(n, m);`

```
This usage is syntactically correct since it is reassigning the string
variable message to it's erased and modified form. Although this is a
correct usage, it is unnecessary to reassign message. A simpler usage is
shown in part (a).
```

c. (2 points) `cout << message.find(n);`

```
This usage is syntactically correct since the find method will convert the
characters inside of the strings into integers and then it will search the
string for the integer n.
```

d. (2 points) `message.size() = n;`

```
This is not correct usage because message.size() is not a variable and it is
not assignable.
```

e. (2 points) `cout << message.rfind("x");`

```
Yes, this is correct usage of the rfind() method because it is searching the
string message backwards for the substring consisting of a single character
"x". It prints out without error because the method rfind() returns either
string::npos if not found or the last index of the occurance of "x".
```

2. (10 points) The following code takes in a string input from the user and performs an integer multiplication, as seen in the example run here. Note that the input string will contain the asterisk character `'*'`:

```
Enter 2 integer numbers to be multiplied, like this: num1*num2: 15*3
The answer is: 45
```

Complete the missing code below that performs this task (it can be done in 2 lines, but you can use more if you like).

```
string s; int k(0);
cout << "Enter 2 integer numbers to be multiplied, like this: num1*num2: ";
cin >> s;

int num1 = stoi(s.substr(0,s.find('*'))), num2 = stoi(s.substr(s.find('*')+1));
k = num1*num2;

cout << "The answer is: " << k << endl;
```

3. (20 points) Given the declaration of a C-string variable, where `MAX` is a defined constant: `char buffer[MAX];`

The C-string variable `buffer` has previously been assigned in code not shown here. For correct C- string variables, the following loop reassigns all positions of `buffer` the value ‘z’, leaving the length the same as before. Assume this code fragment is embedded in an otherwise complete and correct program. Answer the questions following this code fragment:

```
int index = 0;
while (buffer[index] != '\0') {
   buffer[index] = 'z';
   index++;
}
```

a. (10 points) Explain how this code can destroy memory beyond the end of the array.

```
Essentially this loop can destroy memory if the loop does not terminate
correctly (as expected by the '\0') and thus the loop will not stop.
Therefore, this loop can continue to insert the character 'z' into indices
```

```
that are past what the buffer array has actually made space for by the
declaration of buffer as 'char buffer[MAX];'.
```

b. (10 points) Modify this loop to protect against inadvertently changing memory beyond the end of the array.

```
int index = 0;
while (buffer[index] != '\0' && index < MAX)
{
    buffer[index] = 'z';
    index++;
}
```

4. (20 points) Show the output produced when the following code (entire program not shown) executes. *If there is an error in this code*, point it out and explain why it is not correct. You are encouraged to also try to compile this to verify your results.

```
string name = "Porcupine Tree";
cout << "NAME = " + name << endl;
cout << name.length() << endl;
name.erase(8, 6);
cout << name << endl;
name.append("Dean WD Morgan");
cout << name << endl;
name.insert(22, "@TWD");
cout << name << endl;
name.replace(23, 3, "The WD");
cout << name << endl;
cout << name.find("WD") << endl;
cout << name.rfind("WD") << endl;
cout << name.rfind("cupi") << endl;
for (int i = name.length(); i > 20; i--) {
   cout << name[i-1];
   cout << endl;
}
```

No errors in this code.
Output:

```
NAME = Porcupine Tree
14
Porcupin
PorcupinDean WD Morgan
PorcupinDean WD Morgan@TWD
PorcupinDean WD Morgan@The WD
13
27
3
D
W

e
h
T
@
n
a
```

5. (20 points) Write the full definition of a function called `FunString()` that takes a string argument and does 2 things: (1) it prints the *second* half of the string backwards (while still printing the first half normally), and (2) it reports on how many words the original string has (assume a word is separated with space characters). For example, if the argument is “All the strings”, the function should print out “All thesgnirts” on one line and then the number `3` on the next line.

```
void FunString(string text)
{
    int halfLength = text.length()/2;
    //print out first half of the string in order
    for (int i=0;i<halfLength;i++)
    {
        cout << text[i];
    }

    //print out the second half of the string in backwards order
    for (int j=text.length()-1;j>=halfLength;j--)
    {
        if (j>halfLength)
        {
            cout << text[j];
        }
        else
        {
            cout << text[j] << endl;
        }
    }

    int spaces=0;
    //count the number of spaces in the string
    for (int k=0;k<text.length();k++)
    {
        if (text[k]==' ')
        {
            spaces++;
        }
    }

    cout << spaces+1 << endl;
}
```

6. (20 points) Write a full definition for a function called `IsLoud()` that takes in a string argument and checks if each character in the string is an uppercase character *or* a `'!'`. If all characters pass this test, then the function returns true, otherwise it returns false.

```
bool IsLoud(string text)
{
    /*
    takes in a string argument and checks if each character in the string is
    an uppercase character or a '!'.
    If all characters pass this test, then the function returns true,
    otherwise it returns false.
    */

   //assume true until proven false
   bool isGood = true;
   for (int i=0;i<text.length();i++)
   {
       if (text[i] != '!' && (text[i] < 'A' || text[i] > 'Z'))
       {
            isGood = false;
            break;
       }
   }
   return isGood;
}
```