CS 171 Computer Programming 1

Winter 2016

Lab 8 – Strings

NAME: Joseph Mulray LAB: 8

Question 1: (3pts)

For the first part of this week's lab, you are going to build a program that performs the same kind of search and replace operation that virtually all text editors provide. The first step in developing the program is to write a main function with the following characteristics:

- Read a line of input to a variable called `pattern`
- Read a line of input to a variable called `replacement`
- Read lines of input, stopping when a line consisting of only $$ is read
    - As long as the input is not $$, print the input back out.

NOTE: You will want to allow the user to read in multiple words. `cin` only allows us to read in one word at a time. Do you remember how to read in lines from the command line?

Here's a sample run:

```
What's the pattern you want to look for? Hi
What would like to replace that pattern with? Go
Enter a line to replace in.  Enter $$ to quit: This is going well
This is going well

Enter a line to replace in.  Enter $$ to quit: hi to you
hi to you

Enter a line to replace in.  Enter $$ to quit: $$
```

Copy your main function below.

Answer:

```
// Joseph Mulray

#include <iostream>

#include <string>

using namespace std;
```

```
int main()
{
string pattern, replacement, input;


cout<<"What's the pattern you want to look for? ";
getline(cin, pattern);


cout<<"What would like to replace that pattern with? ";
getline(cin, replacement);

while(input!="$$");
{


cout<<"Enter a line to replace in.  Enter $$ to quit: ";
getline(cin, input);
cout<<input<<endl;

}
   return 0;




}
```

}

Question 2: (5pts)

To your program, add a function that takes *three string arguments*: the pattern, the replacement, and a line of input. *The function will return a string*. If the pattern appears in the line of input, the function should return the empty string. Otherwise, it should return the input line.

For example, if we called our function `strrep` and used it as follows

```
string result  = strrep("hi", "go", "This is going well");
```

`result` would now have the value "".

However, if we called our function with

```
string result  = strrep("bye", "go", "This is going well");
```

it would return "This is going well".

Admittedly we don't actually need the second parameter at this time, but we'll keep it there for use in later questions.

Also change your main function so that it passes the pattern, replacement, and current line of input (unless it was $$) to this function and prints out its result.

Here's a sample run:

```
What's the pattern you want to look for? hi
What would like to replace that pattern with? Go
Enter a line to replace in.  Enter $$ to quit: This is going well
```

```
Enter a line to replace in.  Enter $$ to quit: Bye to you
Bye to you

Enter a line to replace in.  Enter $$ to quit: $$
```

After you wrote your function and tested it via various calls in your main, copy and paste *just your new function* below.

Answer:

```
// Joseph  Mulray

#include <iostream>

#include <string>

using namespace std;


string result( string pattern,string replacement,string input)

{

   if(input.find(pattern)==0)

   {

      return(" ");

   }

   else

   {

      return(input);

   }

}

int main()

{

string pattern, replacement,final, input;


cout<<"What's the pattern you want to look for? ";
```

```
getline(cin, pattern);



cout<<"What would like to replace that pattern with? ";
getline(cin, replacement);



do
{
cout<<"Enter a line to replace in.  Enter $$ to quit: ";
getline(cin, input);
final = result(pattern, replacement, input);
cout<<final<<endl;
}while(input!="$$");
   return 0;



}
```

Question 3 (5pts)

For the next step, modify your new function to change the behavior when the pattern is found in the input line. If the pattern is found, your function should return a string that is the input line with the pattern deleted.

For example, if the pattern is "abc" and the input line is "123abc456" the function should return the string "123456".

Again, after changed your function and tested it via various calls in your main, copy and paste *just your updated function* below.

Answer:

```
// Joseph Mulray

#include <iostream>

#include <string>

using namespace std;

string result(string input,string pattern,string replacement)
{
   int position ;
   position = input.find(pattern);
   if(position==-1)
   {
      return("");
   }
   else
   {
      input.replace(position, pattern.length(),"" );
```

```
        return input;
    }

}

int main()

{

string pattern, replacement,final, input;

cout<<"What's the pattern you want to look for? ";

getline(cin, pattern);

cout<<"What would like to replace that pattern with? ";

getline(cin, replacement);

cout<<"Enter a line to replace in.  Enter $$ to quit: ";
getline(cin, input);



while(input!="$$")
{

final = result(input, pattern, replacement);
```

```
cout<<"FINAL: "<<final<<endl;


cout<<"Enter a line to replace in.  Enter $$ to quit: ";
getline(cin, input);


}


   return 0;


}
```

Question 4 (8pts)

The final step in the development of this program is to change the string processing function so that it now returns a string containing the input line with the pattern replaced by the replacement.

For example, if the pattern is "dog", the replacement is "cat", and the input line is "dog food", your function should return "cat food".

When you have the program working correctly, upload your source code here.

Answer:

```
// Example program


#include <iostream>


#include <string>


using namespace std;


string result(string input,string pattern,string replacement)
{
   int position ;
   position = input.find(pattern);
   if(position==-1)
   {
      return("");
   }
   else
   {
      input.replace(position, pattern.length(), replacement);
      return input;
   }


}
```

```
int main()

{

string pattern, replacement,final, input;

cout<<"What's the pattern you want to look for? ";

getline(cin, pattern);

cout<<"What would like to replace that pattern with? ";

getline(cin, replacement);

cout<<"Enter a line to replace in.  Enter $$ to quit: ";
getline(cin, input);



while(input!="$$")
{

final = result(input, pattern, replacement);
cout<<"FINAL: "<<final<<endl;

cout<<"Enter a line to replace in.  Enter $$ to quit: ";
getline(cin, input);
```

```
}

    return 0;

}
```

Question 5 (5pts)

A common children's word game is the "language" Pig Latin. It is constructed by stripping the initial consonant sound from a word, and appending a new syllable made from the consonant sound and the vowel sound **"**`ay`**" .**

Words with no initial consonant sound are treated as though they had an initial consonant of "w".

```
Ouyay illway itewray away ogrampray otay anslatetray intoway Igpay Atinlay.
```

We'll build a program over Questions 5-7 to take print out pig latin versions of entered text!

For now, your main function should read from the input, one **word** at a time until it gets the "word" `$$`.

For each word, call a function (which you will also have to build) called `isvowel` on the first **letter** of each word. *Print out those words for which `isvowel` returns true.*

In the answer area copy your entire program (which for now should contain at least your main function and the isvowel function)

Answer:

```
// Joseph Mulray

#include <iostream>

#include <string>

using namespace std;


bool isVowel(string word)

{

   char letter = word [0];


  switch(letter)

{

  case 'a':

  case 'A':
```

```
  return true;
  break;
  case 'e':
  case 'E':
  return true;
  break;
  case 'i':
  case 'I':
  return true;
  break;
  case'o':
  case 'O':
  return true;
  break;
  case'u':
  case 'U':
  return true;
  break;
  default:
  return false;
  break;
}



}
int main()
{
```

```
string word;
bool isTrue;



cout<<"Enter a word, Enter $$ to quit: ";
cin>>word;

while(word!="$$")
{
   isTrue =isVowel(word);

   if(isTrue==true)
   {
      cout<<word<<endl;
   }

   cout<<"Enter a word, Enter $$ to quit: ";
   cin>>word;

}


}
```

Question 6 (5pts)

Step 2 of this program is to append the suffix "way" to each of the words that begins with a vowel.

For example, the word "`if`" will become "`ifway`".

You will continue to not print words that begin with a consonant.

After you have tested your program, just copy some of its outputs below (we won't ask for the code this time, though we'll see it later).

Answer:

```
Enter a word, Enter $$ to quit: if
 ifway
Enter a word, Enter $$ to quit: auto
 autoway
Enter a word, Enter $$ to quit: word
Enter a word, Enter $$ to quit: $$
```

Question 7 (8pts)

The final step in this program is to process those words that begin with a consonant.

If a word starts with a consonant, then initial consonants (all of them until you hit a vowel) should be moved to the end of the word and the string "ay" appended after that.

Some examples:

- "pig" → "igpay"
- "banana" → "ananabay"
- "trash" → "ashtray"

It will probably be easiest to create another function that manipulates words that begin with consonants.

Here's a sample run for the final program:

```
Enter a word.  Enter $$ to quit: pig
igpay

Enter a word.  Enter $$ to quit: banana
ananabay

Enter a word.  Enter $$ to quit: trash
ashtray

Enter a word.  Enter $$ to quit: $$
```

When your program works correctly, copy your entire program (the main and all the other functions) to the answer area below:

Answer:

```
// Joseph Mulray
#include <iostream>
#include <string>
using namespace std;


void isConsonant(string word)
{

    cout<<word.substr(1,word.length()) + word[0] + "ay"<<endl;

}


bool isVowel(string word)
{
    char letter = word [0];

   switch(letter)
{
   case 'a':
   case 'A':
   return true;
   break;
   case 'e':
   case 'E':
   return true;
   break;
   case 'i':
   case 'I':
   return true;
   break;
   case'o':
   case 'O':
   return true;
   break;
   case'u':
   case 'U':
   return true;
   break;
   default:
   return false;
   break;
}


}
int main()
{
string word;
```

```
bool isTrue;


cout<<"Enter a word, Enter $$ to quit: ";
cin>>word;

while(word!="$$")
{
    isTrue =isVowel(word);

    if(isTrue==true)
    {
        cout<<word + "way"<<endl;
    }
    else
    {
        isConsonant(word);
    }

    cout<<"Enter a word, Enter $$ to quit: ";
    cin>>word;

}

}
```

Question 8 (8pts)

Your final problem for this week's lab is to write a program which *only outputs palindromes*.

Palindromes are strings that are the same when scanned from left to right as when scanned from right to left.

For example, `"abba", "x", "abc12321cba"` are all palindromes.

For this program, you will input a single word. If it is a palindrome, print it out. If it is not a palindrome, then make a palindrome from it by reversing it and appending the reverse string to the original.

Some input-output pairs will include:

- `abba→abba`
- `abc→abccba`

- x → x
- xx → xx
- Drexel → DrexellexerD

Here's how those tests would like in your final running program:

```
Enter a word.  Enter $$ to quit: abba
abba

Enter a word.  Enter $$ to quit: abc
abccba

Enter a word.  Enter $$ to quit: x
x

Enter a word.  Enter $$ to quit: xx
xx

Enter a word.  Enter $$ to quit: Drexel
DrexellexerD

Enter a word.  Enter $$ to quit: $$
```

When your program works correctly, copy your entire program (the main and any other functions you decide to make) to the answer area below:

Answer

//Joseph Mulray

```
#include <iostream>

#include <string>

using namespace std;


void palindromes(string word)

{

string str;


for(int x=word.length()-1; x>=0;x--)
```

```
{
   str+= word[x];
}

if(word==str)
{
   cout<<word<<endl;
}
else
{
   cout<< word + str << endl;
}

}




int main()
{
   string word;

   cout<<"Enter a word, Enter $$ to quit: " ;
   cin>>word;

   while(word!="$$")
   {
   palindromes(word);
```

```
    cout<<"Enter a word, Enter $$ to quit: " ;
    cin>>word;

    }


}
```