Tuesday, 07-08-2014

SWE315 : C++

**Lesson 9 - MidTerm dry-run ( Due date 07-15 : a Week!!)**

Notes:

- **Open material** - Books, homework. Internet just for looking into reference material.
  - Example websites: http://www.cplusplus.com/reference/
  - For ++ for example: http://en.cppreference.com/w/cpp/language/operators or http://www.tutorialspoint.com/cplusplus/increment_decrement_operators_overloading.htm
- **Timed exam** – If you need accommodation, let me know beforehand.
- **In the real exam**:
  - There will also be (2 questions of) "What this program does" questions (not too tricky)
  - The main question will NOT be on strings…

Solution:

1. Please send solution to: zbaharav@cogswell.edu
2. You know the drill by now: Simply hit reply, and no zipped directories etc..
3. You will need to attach this word document, where at the bottom (see place for this) you paste
   a. StringPig.h
   b. StringPig1.cpp
   c. Screen shots of the console output.

====

(see also hints-and-guidelines below!)

Create a class called *StringPig1* , which is a basic conversion of a string into PigLatin version (http://en.wikipedia.org/wiki/Pig_Latin ) . It converts correctly words which start with vowels. The class should have the following variables and interfaces:

a. Variables:
   i. Character pointer *str* .
   ii. Integer variable *len* (holding the length of str, including terminating null character).
b. Interfaces:
   i. Constructor :
      1. Empty constructor
      2. Constructor from a character string:

```
StringPig1(const char *s);
```

      3. Copy constructor:

```
StringPig1(const StringPig1& s);
```

4. Assignment operator:

```
StringPig1& operator=(const StringPig1& s);
```

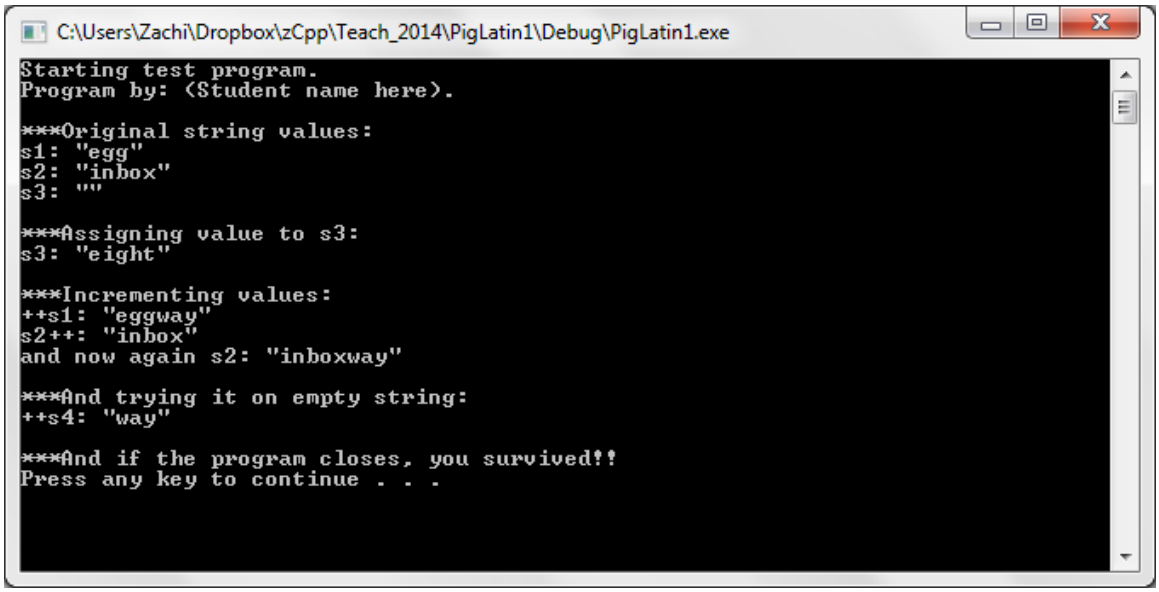
ii. Destructor, to clear memory allocated.

iii. Overload the following operators:

1. <<
2. ++ (prefix)
3. ++ (postfix) <-- This is MORE challenging, so you may wish to leave this for the very end.

Please use the following test-program to validate your solution, and produce the screen shot as below. (you will need to change the 'student name').

Screen shot :




Test program:

```
// main.cpp
// Test file for StringPig1.cpp

#include <iostream>
#include "StringPig1.h"

using std::cout;
using std::endl;

int main(void)
{

	{ // This is meant so we can see if the destructor 'kills' the program
```

```
        StringPig1 s1("egg");
        StringPig1 s2 = "inbox";
        StringPig1 s3;

        cout << "Starting test program.\n";
        cout << "Program by: (Student name here).\n";
        cout << "\n***Original string values:\n";

        cout << "s1: \"" << s1 << "\"\n";
        cout << "s2: \"" << s2 << "\"\n";
        cout << "s3: \"" << s3 << "\"\n";

        cout << "\n***Assigning value to s3:\n";
        s3 = "eight";
        cout << "s3: \"" << s3 << "\"\n";

        cout << "\n***Incrementing values:\n";
        cout << "++s1: \"" << ++s1 << "\"\n";
        cout << "s2++: \"" << s2++ << "\"\n";
        cout << "and now again s2: \"" << s2 << "\"\n";

        cout << "\n***And trying it on empty string:\n";
        StringPig1 s4;
        cout << "++s4: \"" << ++s4 << "\"\n";

    }

    cout << "\n***And if the program closes, you survived!!\n";
    system("pause");

    return 0;
}
```

Hints and guidelines:

1. You should know how to create the simple class with variables and methods.
2. You will need to dynamically allocate a 1D array.
3. Remember to clear this space in the destructor.
4. Important:
    a. Implement the simple things first.
    b. Comment stuff from main.cpp when you only implement parts.
    c. Only after you did well with the assignment operator, head into the ++ overloading.

=== End of DryRun ====

Solution:

```
// StringPig1.h

#ifndef _STRINGPIG1_H_
```

```
#define _STRINGPIG1_H_

#define _CRT_SECURE_NO_WARNINGS

#include <iostream>

class StringPig1
{
private:
	char *str;		// character array
	int len;		// length

public:
	// Constructors
	StringPig1();
	StringPig1(const char *s);
	StringPig1(const StringPig1& s);		// Copy constructor
	StringPig1& operator=(const StringPig1& s);	// Assignemnt operator

	// Destructor
	~StringPig1();

	// Overloaded
	friend std::ostream& operator<<(std::ostream& os, const StringPig1& st);
	StringPig1 operator++();		// ++ prefix
	StringPig1 operator++(int); // postfix ++

};

#endif
```

# // StringPig1.cpp

```
#include "StringPig1.h"
#include <cstring>

// Constructors
// Default constructor
StringPig1::StringPig1()
{
	len = 0;
	str = new char[len+1];
	std::strcpy(str,"");
}

// FromString constructor
StringPig1::StringPig1(const char *s)
{
	len = std::strlen(s);
	str = new char[len+1];
	std::strcpy(str,s);
}

// Copy constructor
StringPig1::StringPig1(const StringPig1& s)
```

```
{
	len = s.len;
	str = new char[len+1];
	std::strcpy(str,s.str);
}

// Asignment operator
StringPig1& StringPig1::operator=(const StringPig1& s)
{
	if (this == &s)
		return *this;
	delete [] str;
	len = s.len;
	str = new char[len+1];
	std::strcpy(str,s.str);
	return *this;
}

// Destructor
StringPig1::~StringPig1()
{
	delete [] str;
}

// Overloaded
std::ostream& operator<<(std::ostream& os, const StringPig1& st)
{
	os << st.str;
	return os;
}

// ++ prefix
// Adds "way" to the string
StringPig1 StringPig1::operator++()
{
	// Many ways to do it.
	// We are doing the most explicit one (no shortcuts)

	// Save a copy first!
	char* tmp;
	tmp = new char [len+1];
	std::strcpy(tmp,str);

	// Clear the old one
	delete [] str;

	// Now set space for the longer one
	len = len+3;
	str = new char[len+1];
	// copy back
	std::strcpy(str,tmp);
	// concatenate
	std::strcat(str,"way");

	return *this;
}
// postfix ++
// It is important to note that here you CANNOT use the reference
```

```
// (and that's why we removed it from the prefix as well
StringPig1 StringPig1::operator++(int)
{
	// save original value
	StringPig1 S(*this);

	++(*this);

	return S;
}
```

=== END of Solution