CIS 15 Spring 2009, Assignment VI

Instructions

- This is the assignment for Unit VI.
- It is worth 8 points.
- **It is due on Wednesday May 20th** and must be submitted by email (as below).
- **Follow these emailing instructions:**

  1. Create a mail message addressed to **parsons@sci.brooklyn.cuny.edu** with the subject line **cis15 hw6**.
  2. Attach ONLY the files I specify below
  3. Write your name, that is the name under which you registered for the course, in the email. When I get an email from `deathmetal@aol.com` or `pinkprincess@yahoo.com`, I can usually guess whose program it is, but that is not as good as *knowing* whose program it is.
  4. Failure to follow these instructions will result in points being taken away from your grade. The number of points will be in proportion to the extent to which you did not follow instructions . . . (which can make it a lot harder for me to grade your work)

Description

This assignment has two parts:

1. You will write a program with two functions that help you explore **recursion**.
2. You will write three small programs that help you explore **templates** and the **STL**.

A. Printing a string

Create a file called **hw6-1.cpp**. In it, write a program that prompts the user to enter a string, reads the string and stores it as an array of `char`, then displays it, one character at a time.

Design requirements:

- Create a class called `mystring`, which contains a private member that is an array of `char`, a constructor, and a recursive function called `print()`.
- You MUST use recursion for `print()`!
- If you want, you may create separate files **mystring.cpp** and **mystring.h** for defining the class, or you can keep it all in one file—that's up to you.

Compile, link and run your code. Test it to make sure it works robustly.

B. Printing the string backwards

Modify **hw6-1.cpp** to include another recursive function in your class called `printback()` that prints out the string backwards. In other words, if I enter the string: `HELLO`, then the program should output: `OLLEH`.

- As above, you MUST use recursion for printing the string.

Compile, link and run your code. Test it to make sure it works robustly.

C. Writing your own template

1. Create a file called **hw6-2.cpp**.
2. In it, put the following header definition:

```
template<class TYPE>
class Elements {
public:
  Elements( int size=1 );
  ~Elements();
  void set( int index, TYPE value );
  TYPE get( int index );
  int getSize();
private:
  TYPE *data;
  int size;
}; // end of class Elements
```

3. Add the following main function definition:

```
int main() {
  string tmp = "hello";
  Elements<char> A(5);
  for ( int i=0; i<tmp.size(); i++ )
    A.set( i,tmp[i] );
  for ( int i=0; i<A.getSize(); i++ )
    cout << A.get( i ) << " ";
  cout << endl;
} // end of main()
```

4. Complete the template class by writing definitions for the 5 functions:
   - `Elements( int size=1 );`
   - `~Elements();`
   - `void set( int index, TYPE value );`
   - `TYPE get( int index );`
   - `int getSize();`
5. Compile, link and run your code.
6. Test it to make sure it works robustly. The output should look like this:

```
h e l l o
```

D. Using the STL `list` class

1. Create a file called **hw6-3.cpp**.
2. Copy the `main()` function from **hw6-2.cpp** into your new file (hw6-3.cpp).
3. Modify it so that instead of using the `Elements` class that you created yourself, it is using the STL `list` class.
4. Inside the `for` loop for inserting elements onto the list, you will need to use either the `list` function named `push_front()` or `push_back()`.

5. Inside the `for` loop for the printing elements that are on the list, you will need to call either `back()` (to access the last element on the list) and `pop_back()` (to remove the last element from the list) or `front()` and `pop_front()` (to access and then remove the first element on the list).

6. Refer to the documentation handed out in class on the `list` template. This can also be found online at:

   `http://www.cppreference.com/cpplist/index.html`

7. Compile, link and run your code.

8. Test it to make sure it works robustly. The output should look the same as it does with the first program (hw7a).

E. Using the STL `list` class with an `iterator`

1. Create a file called **hw6-4.cpp**.

2. Copy your code from **hw6-3.cpp** into your new file (hw6-4.cpp).

3. In the second `for` loop, where you print the elements that are on the list, replace the use of `i` with an **iterator**.

   Refer to the class notes (lecture VI.2) and the textbook (chapter 7) for help with iterators.
   You can also find help online at:

   `http://www.cppreference.com/iterators.html`

4. Compile, link and run your code.

5. Test it to make sure it works robustly. The output should look the same as it does with the first template program (hw6-2).

F. Submission

Create a ZIP file that includes **hw6-1.cpp** (or, optionally **mystring.cpp** and **mystring.h**), **hw6-2.cpp**, **hw6-3.cpp** and **hw6-4.cpp**, and email this to me.

G. Marking rubric

- Part A: *1.5 points*
  1 point for correctly defining a `mystring` object
  0.5 points for the `print` function.
- Part B: *0.5 points*
- Part C: *2.5 points*
  0.5 points for each of the 5 functions
- Part D: *2 points*
  1 points for correctly defining a `list` object
  0.5 points for correctly adding items to the list
  0.5 points for correctly printing the list
- Part E: *1.5 point*
  1 point for correctly defining a list iterator
  0.5 points for correctly using the list iterator to print out the list contents