CS2134 HOMEWORK 4*
Spring 2016
Due 11:00 p.m. March 1, 2016

Be sure to include your name at the beginning of each file! Assignment 4 include a programming portion and a written part. The programming portion must compile and consist of a single file (hw04.cpp). The written portion should consist of a single file (hw04written) in a .pdf format. Be sure to include your name at the beginning of each file! You must hand in the file via NYU Classes.

Programming Part: Some of the recursion programming exercises could easily be written without recursion, but in order to practice recursion, you will write them recursively.

1. Create a menu for the user to access the data created in programming problem 1 in assignment 3B[1]

   In the interactive phase, the program will repeatedly prompt the user to take any one of the following actions:

   (a) Print out the information about all the train stops on a specific route; if no route is found, you should print out a message informing the user.

   (b) Print out the information about a specific train stop; if no train stop is found, you should print out a message informing the user.

   (c) Print out all the train stops within a certain distance, where the user enters the geographical coordinates. If no stop is found within the distance, print out a message informing the user.

   (d) quit

   To perform these operations you will define one generic function template[2] and four functors.

   Write a generic function template called `perform_if` that takes four parameters: two iterators, `itrStart, itrEnd`, and two functors, `pred` and `op`. The two iterators should have have the capability of forward iterators. Both functors will have the overloaded `operator()` that takes one argument. The functor `pred`'s overloaded `operator()` returns a boolean value; the functor `op`'s overloaded `operator()` does not return any value. The function will apply `pred`'s overloaded `operator()` to each item in the range `[itrStart, itrEnd)`. If `pred`'s overloaded `operator()` returns `true`, the other functor `op`'s overloaded `operator()` is then applied to the item, e.g.

```
if ( pred( *itr ) )
{
    op(*itr);
    how_many++;
}
```

*10% extra credit will be given if you turn this assignment in on Monday February 29

[1]You may use the published solution as long as you cite that you are using it.

[2]This generic function template should be written in the STL generic template function style and should *not* written to be used only with the train stop data.

The return value of the generic function template `perform_if` is an `int` which returns the number of items for which the functor `pred` returned `true`.

The four functors you will write are:

(a) Write a functor called `isStopOnRoute` that has a private member variable called `route` of type `char`. The constructor takes a single parameter of type `char` which it uses to initialize the variable `route`. It also contains an overloaded `operator()` that takes a single parameter of type `trainStopData`; the operator returns `true` if the train stop is on the route.[3]

(b) Write a functor called `isSubwayStop` that has a private member variable called `stopId` of type `string`. The constructor takes a single parameter of type `string` which it uses to initialize the variable `stopId`. It also contains an overloaded `operator()` that takes a single parameter of type `trainStopData`; the operator returns `true` if the train stop has an id which is the same as `stopid.`

(c) Write a functor called `isSubwayStopNearX` that has three private member variables called `longitude, latitude`, and `d` of type `double`. The constructor takes three parameters of type `double` which it uses to initialize the private member variables. It also contains an overloaded `operator()` that takes a single parameter of type `trainStopData`; the operator returns `true` if the distance between the train station location and the value of the functors private member variables, `longitude` and `latitude`, is at most `d`. The distance between two points on a sphere is computed using the `haversine` formula. The code to compute this formula is on NYU Classes in a file called `haversine.txt`.[4]

(d) Write a functor called `printTrainStopInfo`. This class contains a single method, an overloaded `operator()`, that takes a single parameter of type `trainStopData` that prints out the train stop information.

2. Write a recursive function called `GCD` that takes as arguments two integers `a, b`, and returns the greatest common divisor of the two numbers by using the following rule.

   - `GCD(a,b)` is a if $b == 0$
   - `GCD(a,b)` is `GCD(b,a)` if $a < b$
   - Otherwise `GCD(a,b)` is `GCD(b, a mod b)`

3. Write a program that sums the items in a vector using a *divide and conquer* algorithm. Create a driver function to call the recursive function; the driver function returns the value returned by the recursive function.

   The driver function takes one parameter of type `vector<int>` and returns an `int`. The prototype for driver function is:

```
int sumVector( const vector<int> & a) // driver program
```

   The recursive function's parameters should be iterators that signify a range `[left, right)`.

   Remember to include a base case.

   If a contains $\{-9, 12, 8\}$ then the function returns 11. (i.e. $11 = -9 + 12 + 8$.)
   If a contains $\{3\}$ then the function returns 3.
   If a contains $\{-21, 31, 14, 1, -20\}$ then the function returns 5. (i.e. $5 = -21 + 31 + 14 + 1 - 20$.)

[3] The train stop route is the first letter of the train stop id. For example, train stop id 202 is on route 2, and train stop id $B20$ is on route B.

[4] You might need to add `#define _USE_MATH_DEFINES.`

The function should:
Divide the vector in half.
Recurse on the left hand side.
Recurse on the right hand side.
Return the sum of both sides.

This function should not have any loops.

4. (Extra Credit Problem) Problem 8.25 from the older Weiss book:

   "Write the routine with the declaration:

   ```
   void permute( const string & str );
   ```

   that prints all the permutations of the characters in the `str`. If `str` is `"abc"`. then the strings output are `abc, acb, bac, bca, cab,` and `cba`. Use recursion."[5]

[5]Think of using the function permute as a driver routine.

Written Part:

1. In programming part 1 of this assignment, you are asked to write a generic function template called `perform_if`.

   - Write the pseudo code for the three to six main steps need to implement `perform_if`.
   - Write the `preconditions` and `postconditions` of the function.
   - Using Big-Oh notation, what is the running time of this generic function template?

2. Show the recursion tree and the runtime for:

   (a) programming question 2 when $a = 30$ and $b = 42$. In your recursion tree show the parameters a and b.

   (b) programming question 3 when $a = \{1, 2, 3, 4\}$. In your recursion tree show the items in the range $[left, right)$.

3. Given the following code snippet:

```
vector<int> a  {1,2,3,4, ..., n}; // vector, a, has n items
vector<int>::iterator itrStart;
vector<int>::iterator itrMid;
vector<int>::iterator itrEnd;
```

   Assign values to the iterators, `itrStart, itrMid, itrEnd`, so that:

   (a) `[itrStart, itrMid)` refers to the range `1`,2, 3, ..., n/2

   (b) `[itrMid, itrEnd)` refers to the range `n/2+1`,n/2+2, ..., n

4. (a) What is printed by the following function call: `myRecFunc(4)`.

   (b) What is the running time of `myRecFunc(n)`.

```
void myRecFunc(int n)
{
   cout << n << ": ";
   if (n < 1) return;
   myRecFunc(n/2);
   myRecFunc(n/2);
   for (int i = 1; i < n; ++i)
      cout << "*";
   cout << endl;
}
```

5. For this recursive Fibonacci function, `fib`, how many function calls are made if n=3; how about if n=4? Using the the number of function calls `fib` made when when n=3 and when n= 4, compute how many function calls are made when n=5. **Show your work.**

```
int fib( int n )
{
    if( n <= 1 )
        return 1;
    else
        return fib( n - 1 ) + fib( n - 2 );
}
```