CSCI 241 Assignment 8, Part 3
100 points

Assignment

For this final part of the assignment you will write several C++ template functions to sort a list of items using the recursive **merge sort** algorithm.

Program

Implement the following template functions in a header file called `mergesort.h`. This header file should have header guards (as usual) and should contain both the prototypes and definitions for the functions.

- `template <class T>`
  `void mergeSort(vector<T>& set, bool (*compare)(const T&, const T&))`

  This function should sort the items in the vector `set` using the merge sort algorithm. The first argument to this function is a reference to a `vector` object containing the list of items to sort. The second argument is a pointer to a comparison function that can be used to compare two items of the template type.

  This function should call the recursive merge sort function, passing it the vector, the subscript of the first vector element (which is 0), the subscript of the last vector element (which is `set.size() - 1`), and the pointer to the comparison function (`compare`), e.g.:

  ```
  mergeSort(set, 0, set.size()-1, compare);
  ```

- `template <class T>`
  `void mergeSort(vector<T>& set, int low, int high, bool (*compare)(const T&, const T&))`

  This recursive function divides a vector into two subvectors, sorts them, and then merges the two sorted subvectors.

  ```
  int mid;

  if (low < high)
     {
     mid = (low + high) / 2;

     // Divide and conquer

     mergeSort(set, low, mid, compare);
     mergeSort(set, mid+1, high, compare);

     // Combine
     merge(set, low, mid, high, compare);
     }
  }
  ```

- `template <class T>`
  `void merge(vector<T>& set, int low, int mid, int high, bool (*compare)(const T&, const T&))`

  This function merges two sorted subvectors.

```
// Create temporary vector to hold merged subvectors
vector<T> temp(high - low + 1);

int i = low;      // Subscript for start of left sorted subvector
int j = mid+1;    // Subscript for start of right sorted subvector
int k = 0;        // Subscript for start of merged vector

// While not at the end of either subvector
while (i <= mid && j <= high)
   {
   if (compare(set[j], set[i]))
      {
      Copy element j of set into element k of temp
      Add one to j
      Add one to k
      }
   else
      {
      Copy element i of set into element k of temp
      Add one to i
      Add one to k
      }
   }

// Copy over any remaining elements of left subvector
while (i <= mid)
   {
   Copy element i of set into element k of temp
   Add one to i
   Add one to k
   }

// Copy over any remaining elements of right subvector
while (j <= high)
   {
   Copy element j of set into element k of temp
   Add one to j
   Add one to k
   }

// Copy merged elements back into original vector
for (i = 0, j = low; j <= high; i++, j++)
   Copy element i of temp into element j of set
```

A driver program, `assign8.cpp`, is provided below to test your code for this part of the assignment. A copy of the driver program can also be found on `turing` at
`/home/turing/t90kjm1/CS241/Code/Fall2016/Assign8/Part3/assign8.cpp`.

```
/*****************************************************************
   PROGRAM:    CSCI 241 Assignment 8
   PROGRAMMER: your name
   LOGON ID:   your z-ID
   DUE DATE:   due date of assignment
```

```
   FUNCTION:   This program builds and sorts lists using the quick
               sort and merge sort algorithms.
*****************************************************************/

#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include "sorts.h"
#include "quicksort.h"
#include "mergesort.h"

using std::cout;
using std::fixed;
using std::left;
using std::setprecision;
using std::string;
using std::vector;

// Data files

#define D1 "/home/turing/t90kjm1/CS241/Data/Fall2016/Assign8/data8a.txt"
#define D2 "/home/turing/t90kjm1/CS241/Data/Fall2016/Assign8/data8b.txt"
#define D3 "/home/turing/t90kjm1/CS241/Data/Fall2016/Assign8/data8c.txt"

// Output formatting constants

#define INT_SZ 4    // width of integer
#define FLT_SZ 7    // width of floating-pt number
#define STR_SZ 12   // width of string

#define INT_LN 15   // no of integers on single line
#define FLT_LN 9    // no of floating-pt nums on single line
#define STR_LN 5    // no of strings on single line

int main()
   {
   vector<int> v1;      // vector of integers
   vector<float> v2;    // vector of floating-pt nums
   vector<string> v3;   // vector of strings

   // Print header message
   cout << "*** CSCI 241: Assignment 8 - Output ***\n\n";

   // sort and print first list

   cout << "First list - ascending order:\n\n";
   buildList(v1, D1);
   quickSort(v1, &lessThan);
   printList(v1, INT_SZ, INT_LN);

   v1.clear();

   cout << "\nFirst list - descending order:\n\n";
   buildList(v1, D1);
   mergeSort(v1, &greaterThan);
   printList(v1, INT_SZ, INT_LN);

   // Sort and print second list
```

```
cout << fixed << setprecision(2);

cout << "\nSecond list - descending order:\n\n";
buildList(v2, D2);
quickSort(v2, &greaterThan);
printList(v2, FLT_SZ, FLT_LN);

v2.clear();

cout << "\nSecond list - ascending order:\n\n";
buildList(v2, D2);
mergeSort(v2, &lessThan);
printList(v2, FLT_SZ, FLT_LN);

// Sort and print third list

cout << left;

cout << "\nThird list - ascending order:\n\n";
buildList(v3, D3);
quickSort(v3, &lessThan);
printList(v3, STR_SZ, STR_LN);

v3.clear();

cout << "\nThird list - descending order:\n\n";
buildList(v3, D3);
mergeSort(v3, &greaterThan);
printList(v3, STR_SZ, STR_LN);

// print termination message
cout << "\n*** End of program execution ***\n";

return 0;
}
```

Other Points

- Assignment 8 as a whole is worth 100 points, and only this final part of the assignment needs to be submitted for grading. However, if you are unable to complete all three parts of the assignment, you may submit one of the earlier parts for partial credit.
- As usual, you must have a makefile. The name of your final executable should be `assign8`.
- Don't forget to get rid of all compiler warnings when the `-Wall` compilation option is used.
- As always, programs that do not compile on `turing/hopper` automatically receive 0 points.
- Submit your program using the electronic submission guidelines posted on the course web site and discussed in class.