CSCI 241 Assignment 8, Part 2

Assignment

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

Program

Add the following template functions to your `sorts.h` file. The header file should contain both the prototypes and definitions for these functions.

-
  ```
  template <class T>
  bool lessThan(const T& item1, const T& item2)
  ```

  This function should return true if `item1` is less than `item2` and false if not. You may assume that the data type `T` can be compared using the standard relational operators.

-
  ```
  template <class T>
  bool greaterThan(const T& item1, const T& item2)
  ```

  This function should return true if `item1` is greater than `item2` and false if not. You may assume that the data type `T` can be compared using the standard relational operators.

Implement the following template functions in a header file called `quicksort.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 quickSort(vector<T>& set, bool (*compare)(const T&, const T&))
  ```

  This function should sort the items in the vector `set` using the quick 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 quick 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.:

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

-
  ```
  template <class T>
  void quickSort(vector<T>& set, int start, int end, bool (*compare)(const T&, const
  T&))
  ```

  This function performs the recursive calls to implement the quick sort algorithm. The logic is:

  ```
  int pivotPoint;

  if (start < end)
  ```

```
        {
        pivotPoint = partition(set, start, end, compare);     // Get the pivot point
        quickSort(set, start, pivotPoint - 1, compare);       // Sort first sublist
        quickSort(set, pivotPoint + 1, end, compare);         // Sort second sublist
        }
```

- `template <class T>`
  `int partition(vector<T>& set, int start, int end, bool (*compare)(const T&, const T&))`

  This function selects a pivot element and then partitions the vector around the pivot. The logic is:

  ```
  int pivotIndex, mid;
  T pivotValue;

  mid = (start + end) / 2;

  Swap elements start and mid of the vector

  pivotIndex = start;
  pivotValue = set[start];

  for (int scan = start + 1; scan <= end; scan++)
     {
     if (compare(set[scan], pivotValue))
        {
        pivotIndex++;
        Swap elements pivotIndex and scan of the vector
        }
     }

  Swap elements start and pivotIndex of the vector

  return pivotIndex;
  ```

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/Part2/assign8.cpp`.

```
/*********************************************************************
   PROGRAM:    CSCI 241 Assignment 8, Part 2
   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 algorithm.
*********************************************************************/

#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include "sorts.h"
#include "quicksort.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 - Part 2 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);

   // 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);

   // 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);

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

   return 0;
   }
```