CITY UNIVERSITY OF HONG KONG

Course code & title : EE2331 Data Structures and Algorithms

Session : Semester B, 2013/14

Time allowed : Two hours

This paper has FOUR pages (including this cover page).

---

1. This paper consists of 4 questions.
2. Answer ALL questions in this paper.

This is a closed-book examination.

Candidates are allowed to use the following materials/aids:

Materials/aids other than those stated above are not permitted. Candidates will be subject to disciplinary action if any unauthorized materials or aids are found on them.

Question 1 [20 marks]

Part (a) [10 marks]:

The given program reads temperature values from std::in. For the measurement period, minimum and maximum temperatures are written in std::out. Calculate the time consumption for the given program. Use the O notation. `insert` and `sort` are provided by the STL library. According to STL specification, the running time of `insert` is linear to the input size (i.e. O(n)) and the running time of `sort` is O(nlgn).

```
int main() {

     vector<int> temperatures;
     int tmp = 0;

     // read until end of input sign
     while(cin >> tmp) {

          // add new temperature in the beginning of a vector
          temperatures.insert(temperatures.begin(),tmp);

          // sort
          sort(temperatures.begin(), temperatures.end());

          cout << "Lowest: " << *(temperatures.begin())
               << " Highest: " << *(temperatures.end()-1)
               << " Current: " << tmp << endl;
     }

}
```

Part (b) [10 marks]:

It is clear that the program performance can be improved. Modify the program to make it more efficient. Output of the modified version should be the same as the original version (i.e. after each input value highest, lowest, and current is printed) and values must be stored in a vector. Use the O notation to present the time consumption of the improved version.

Question 2 [20 marks]

Part (a) [8 marks]:

Consider the function `fact(int)` given below. Show the outputs produced by calling `fact(5)`. What is the return value of `fact(5)`?

```
int fact( int n )
{
   if ( n ==1 ) {
      return 1;
   }
   cout << n << " " << n-1 << endl;
   return n * fact( n  - 1 );
}
```

Part (b) [12 marks]:

Design a non-recursive version of the `fact(int)` function in part (a). Printings can be ignored.

Question 3 [20 marks]

Consider a situation that a set of integers are stored in a hash table. The hash table is implemented as a static array size of 12 elements. Collision of elements (several elements with the same hash value) is handled by removing the old hash entry. The hash table is initialized with 12 HashEntry pointers, each initialized to null values. Hash table and HashEntry are declared as follows:

```
HashEntry* hastable[ARRAY_SIZE];

struct HashEntry
{
   int key_;
   int value_;

   HashEntry(int key, int value){
     key_ = key;
     value_ = value:
   }
}
```

Part (a) [10 marks]:
Design a hash function to map a set of integer keys into the given hash table. Show the content of the hash table after inserting the following keys {2, 5, 22, 24, 28, 55} by printing the distribution of keys in a hash table ([key1,..key2,..]).

Part (b) [10 marks]:

Implement the following functions, which are used to store and remove integers in/from the given hash table.

```
void put(int key, int value) //stores the given integer key/value
                                                pair in the hastable

 int get(int key)    //remove element with given key and return
                  its value. If element is not found return -1.
```

Question 4 [40 marks]

Part(a) [10 marks]:

Consider a set of integers represented by a binary search tree. Draw the corresponding binary search tree created as a sequence of insert operations (`insert(int)`). Insertions are performed on the tree in the given input order:

18, 13, 25, 8, 15, 22, 9, 30, 5, 23

Part(b) [15 marks]:

Consider a situation that you need to iterate through the binary tree to reset data value of each node to zero (data = 0). Implement C++ function to iterate all the nodes and reset data (by `visitNode(root)`).

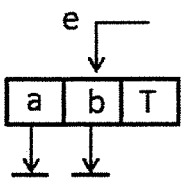
The structure of a tree node is defined below:

```
struct Node
{
   int key;
   int data;
   Node* left;
   Node* right;
}

void visitNode(Node* node)
{
   //iterate all the nodes in binary tree and set data = 0
}
```

Part(c) [15 marks]:

Trie is one type of a tree structure. It is efficient for storing character strings. Assume a Trie structure with alphabet {a,b}. Draw the resulting Trie which stores the following strings {*e*, a, aa, bb, ba, baab}. *e* presents the empty string. Each node is presented as an array of alphabets and a boolean value. Start from the setting shown below (that is a case when Trie only contains empty string *e*).



- END -