CS1020E: Data Structures and Algorithms I

Tutorial 4 – STL, Vector, and ADT

(Week starting 16 February 2015)

1. Vector and Iterator

Given the two code snippets below, check if they are valid. If they are not, justify your answer. If they are, justify your answer and write the output of the code snippet.

a) First Code:

```
vector< int > intVector;
for(int i=0; i!=5; ++i)
  intVector.push_back(i);
vector< int >::iterator myIter = intVector.begin();
for(int j=0; j!=3; j++)
  myIter++;
intVector.erase(myIter);
cout << *myIter << endl; // Quick qn: is this valid?
```

b) Second Code

```
vector< int > intVector;
for(int i=0; i!=5; ++i)
  intVector.push_back(i);
int *myIter = intVector.begin();
intVector.insert(myIter, -1);
print_vector(intVector);
// Assume print_vector will print all element of vector
```

2. Deduplication

Input an integer N followed by an array of integers. Output the array in increasing order without duplicate numbers. For example, if the input is

5

2 3 4 3 2

Your output should be

2 3 4

3. Lucky Numbers

4 and 7 are lucky numbers. If *x* is a lucky number, then so are *4x+7* and *7x+4*. You are given two numbers n and m. You are asked to output all lucky numbers between n and m (inclusive) in increasing order without duplicates. You need to create an ADT called LuckyNumber which has method that takes in n and m and returns the lucky numbers. [Hint: use vector. You solution to the previous question might help with deduplication]

```
#include <iostream>
#include <vector>
using namespace std;

class LuckyNumber {
private:
    vector<int> lucky;
public:
    LuckyNumber() {}
    vector<int> getLuckyNumbers(int n, int m) {
        lucky.clear();
        lucky.push_back(4);
        lucky.push_back(7);
        //Your Codes
        return lucky;
    }
};

int main()
{
    int n, m;
    cin>>n>>m;
    // Your Codes
    return 0;
}
```

4. Deduplication 2

Input an integer *n* followed by *n* strings. Each of the strings contains no more than 100 characters. If string A can be converted to string B after rearrange its characters, we say that A and B are the same (they are anagrams). The strings are case-sensitive. Output the number of distinct strings. For example, if the input is

6
acer
race
wei
jing
gnij
ewi

Your output should be 3, since "acer" and "race" are the same, so are "wei" and "ewi" and "jing" and "gnij".

[Hint: Sorting]