300103 Data Structures and Algorithms

Practical 1 (Week 2)

(2 marks)

Note: *This practical will be marked. You will receive 2 marks if you complete all the tasks correctly during or before your practical class in Week 2. You are highly recommended to complete the tasks at home and show me your solution during your registered practical class in week 2. You will allow one-week delay but will lose one mark for late completion. Your work will be checked and marked face-to-face during the class by your tutor. No offsite submission is accepted.*

Task 1.1

Demonstrate to your tutor that you can use an IDE system for editing and running C++ code either using the lab computers or your own laptops. You can choose either Eclipse, Visual C++, Dev C++, xCode, CodeBlock or other systems with C++ compiler. This task must be done in week 2.

Task 1.2

Consider the definition of the following function template:

```
template <class Type>
void funcExp(Type list[], int size)
{
   Type x = list[0];
   Type y = list[size - 1];
   for (int j = 1; j < size; j++)
  {
      if (x < list[j])
         x = list[j];

      if (y > list[size - 1 -j])
         y = list[size - 1 -j];
   }
   cout << x << endl;
   cout << y << endl;
}
```

Explain to your tutor the meaning and functionalities of this program. Test it with the following data:

```
int list[10] = {5,3,2,10,4,19,45,13,61,11};
string strList[] = {"One", "Hello", "Four", "Three", "How", "Six"};
```

Run the code in an IDE with necessary declarations and a main function to show the output of the following statements?

a. funcExp(list, 10);
b. funcExp(strList, 6);

Task 1.3
Put the ten numbers **0, 1, 2, …, 9** into an array or vector in random order, *i.e*, a random permutation of these numbers, for instance, **4, 2, 0, 9, 6, 5, 7, 1, 3, 8**. Use two different approaches to do that and estimate (informally) which one is possibly more efficient or can't tell. Print the list and calculate how many numbers in the list that are located at their original positions (unchanged positions). For example, in the above list, only 5 is located at its original position (index of 5). Also print out how many calls to *rand()* function.

Task 1.4
Generate twenty random permutations of the number **0, 1, 2, …, 9** using of the algorithms you designed for Task 1.3. Store these permutations into a vector and print them to the screen with the additional information of unchanged positions and number of calls to *rand()*. Calculate the total numbers of unchanged positions. Compare the outcomes of two approaches of random permutation generation.

Hint: The following is an example of output from my code:

```
Outcome of Approach 1:
8 6 4 5 1 0 2 3 7 9 unchanged: 1 random calls to rand(): 100
9 0 4 3 2 6 1 7 8 5 unchanged: 3 random calls to rand(): 100
9 5 3 0 4 8 1 6 2 7 unchanged: 1 random calls to rand(): 100
0 6 3 5 7 8 4 2 1 9 unchanged: 2 random calls to rand(): 100
3 7 9 8 0 4 2 5 1 6 unchanged: 0 random calls to rand(): 100
1 6 0 4 3 2 5 9 7 8 unchanged: 0 random calls to rand(): 100
3 4 0 6 8 2 9 7 1 5 unchanged: 1 random calls to rand(): 100
3 0 4 5 1 9 7 6 8 2 unchanged: 1 random calls to rand(): 100
0 9 1 4 5 2 6 7 3 8 unchanged: 3 random calls to rand(): 100
8 6 9 2 3 5 7 4 0 1 unchanged: 1 random calls to rand(): 100
4 7 9 0 5 2 1 8 6 3 unchanged: 0 random calls to rand(): 100
1 9 5 6 4 7 8 0 2 3 unchanged: 1 random calls to rand(): 100
6 1 2 3 5 4 9 0 8 7 unchanged: 4 random calls to rand(): 100
4 0 9 5 1 8 3 2 7 6 unchanged: 0 random calls to rand(): 100
0 3 1 2 9 7 4 6 5 8 unchanged: 1 random calls to rand(): 100
1 6 0 7 8 5 2 9 3 4 unchanged: 1 random calls to rand(): 100
1 7 8 3 6 2 4 0 9 5 unchanged: 1 random calls to rand(): 100
1 3 6 2 9 4 7 8 5 0 unchanged: 0 random calls to rand(): 100
1 0 4 9 2 5 3 6 8 7 unchanged: 2 random calls to rand(): 100
3 1 6 7 2 9 5 8 4 0 unchanged: 1 random calls to rand(): 100
Total unchanged: 24


Outcome of Approach 2:
9 6 2 8 4 1 5 3 0 7 unchanged: 2 random calls to rand(): 10
8 4 3 5 6 7 2 9 0 1 unchanged: 0 random calls to rand(): 10
9 0 2 6 8 4 1 7 5 3 unchanged: 2 random calls to rand(): 10
0 1 8 7 2 9 3 4 5 6 unchanged: 2 random calls to rand(): 10
7 8 4 0 2 1 9 6 3 5 unchanged: 0 random calls to rand(): 10
```

```
3 5 8 0 4 7 6 2 9 1 unchanged: 2 random calls to rand(): 10
6 3 1 5 0 8 2 7 9 4 unchanged: 1 random calls to rand(): 10
7 3 5 0 2 9 8 4 1 6 unchanged: 0 random calls to rand(): 10
0 2 3 8 7 4 9 5 1 6 unchanged: 1 random calls to rand(): 10
6 3 1 2 8 9 5 7 0 4 unchanged: 1 random calls to rand(): 10
2 8 5 9 1 7 3 4 0 6 unchanged: 0 random calls to rand(): 10
9 4 3 6 1 8 0 5 7 2 unchanged: 0 random calls to rand(): 10
4 5 1 3 2 9 6 8 7 0 unchanged: 2 random calls to rand(): 10
9 6 5 2 1 3 8 7 4 0 unchanged: 1 random calls to rand(): 10
0 2 4 5 9 7 3 1 6 8 unchanged: 1 random calls to rand(): 10
1 0 2 9 4 6 8 5 3 7 unchanged: 2 random calls to rand(): 10
2 4 5 9 1 8 0 6 3 7 unchanged: 0 random calls to rand(): 10
0 5 3 6 4 9 7 8 2 1 unchanged: 2 random calls to rand(): 10
2 5 8 1 6 9 0 3 7 4 unchanged: 0 random calls to rand(): 10
4 7 1 5 9 0 6 8 3 2 unchanged: 1 random calls to rand(): 10
Total unchanged: 20
```