Backup

300103 Data Structures and Algorithms

Practical 1

(Due in Week 2, 2 marks)

Note: *This practical will be marked face-to-face by your tutor during your registered practical session in Week 2. You are required to complete all the tasks before your practical class rather than to do it during class. You are highly recommended to bring your own laptop and demonstrate your work from your laptop to your tutor. No offsite submission is accepted.*

Task 1.1
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 1.2
Write a C++ program (better with a class) to put the six numbers **1, 2, …, 6** into an array of size **nine** in random positions and fill with **0** for the other three positions. For instance, the

following is a possible outcome:

4, 2, 0, 0, 6, 5, 1, 3, 0

where **4** is put in position 0; **2** is put in position 1; **6** is put in position 4; **5** is put in position 5; and so on. You are required to use *rand()* function to randomly generate the position for each number. Add a print function to your class to print the list of the numbers you have generated, including the total number of calls to function *rand()* in the format as follows:

```
2 1 5 4 0 0 6 0 3 , random calls: 11
```

Hint: *Depending on the design of your algorithm, it would need more than six calls to the random generator rand() because some positions might be generated more than once. See **Task** 1.4 for more details.*

Task 1.3

Extend your code for **Task** 1.2 to generate ten lists of the numbers and store them into a *vector*. See the following for an example of outcome:

```
3 0 6 0 2 5 0 1 4 , random calls: 8
4 2 3 0 5 1 0 0 6 , random calls: 8
4 1 3 2 0 0 6 5 0 , random calls: 19
1 3 6 0 2 4 0 0 5 , random calls: 6
0 5 3 2 0 4 6 1 0 , random calls: 9
2 0 0 5 6 1 4 3 0 , random calls: 8
2 1 5 4 0 6 0 3 0 , random calls: 7
4 1 2 6 3 0 0 5 0 , random calls: 8
0 0 6 1 4 3 5 0 2 , random calls: 10
0 3 2 0 5 0 1 4 6 , random calls: 7
```

Task 1.4

Rewrite your code for **Tasks** 1.2 & 1.3 so that the number of function calls to *rand()* to generate each list is exactly six. See the following for an example of outcome:

```
0 0 3 6 2 1 0 5 4 , random calls: 6
0 1 0 0 4 5 3 6 2 , random calls: 6
1 5 0 0 0 3 6 4 2 , random calls: 6
6 3 0 2 5 0 1 0 4 , random calls: 6
5 1 0 0 3 6 2 0 4 , random calls: 6
1 4 6 0 5 0 2 0 3 , random calls: 6
2 0 6 0 4 5 1 3 0 , random calls: 6
2 6 0 4 0 1 3 5 0 , random calls: 6
3 5 6 4 0 0 1 0 2 , random calls: 6
1 6 0 4 0 3 5 2 0 , random calls: 6
```

Note: You are not allowed to use any other functions, which may potentially use random function calls, such as *shuffle()*.