Lab N°01 : Algorithm Analysis

Exercise I.1 (Number of Iterations)

Consider the following two programs, which calculate the sum of the first n natural numbers.

Program (P1.c)

```
#include <stdio.h>
#include <conio.h>

1:  int main (){
2:    int i, n;
3:    long int S = 0;
4:    printf("Enter an integer n :");
5:    scanf("%d", &n);
6:    for ( i=1; i<=n; i++){
7:        S = S + i;
8:    }
9:    printf("\n The sum is: %ld", S);
10:   return 0;}
```

Program (P2.c)

```
#include <stdio.h>
#include <conio.h>

1:  int main (){
2:    int i, j, n;
3:    long int S=0;
4:    printf("Enter an integer n :");
5:    scanf("%d", &n);
6:    for ( i=1; i<=n; i++){
7:        for ( j=1; j<=i; j++){
8:            S = S + 1;
9:        }
10:   }
11:   printf("\n The sum is: %ld", S);
12:   return 0;}
```

Q1) Modify both programs until the number of iterations (**Nbr_it**) in each program has been counted and displayed.

Q2) Fill in the table by running the two programs with different values of n.

N		5	10	50	100	1000	10000	100000
Nbr_It	P1							
	P2							

Q3) Compare the number of iterations of the two programs.

Q4) What do you notice when $n = 100000$ for program **P2**?

Q5) Give the theoretical number of iterations for both programs.

Q6) Compare the results in the table with the theoretical results.

Q7) Give a time complexity relative to n in "Big-Oh"" notation.

Exercise I.2 (Number of Operations & Runtime)

Consider the following two programs, which simultaneously search for the minimum and the maximum.

Program (P1.cc)

```
#include <iostream>
#include<cmath>

using namespace std;

1: int main(){
2:    int i;
3:    long int n, min, max;
      /*Initializations*/
4:    n = 10;
5:    long int *tab=new log int[n];
6:    for(i=0; i<n; i++)
7:       tab[i] = rand()%100;
8:    min = tab[0];
9:    max = tab[0];
10:   for(i=1; i<n; i++){
11:      if (tab[i] > max) max=tab[i];
12:      if (tab[i] < min) min=tab[i];
13:   }
14:   return 0;}
```

Program (P2.cc)

```
#include <iostream>
#include<cmath>

using namespace std;

1: int main(){
2:     int i;
3:     long int n, min, max;
       /*Initializations*/
4:     n = 10;
5:     long int *tab=new log int[n];
6:     for(i=0; i<n; i++)
7:        tab[i] = rand()%100;
8:     if(tab[1]>tab[0]){
9:        min = tab[0];
10:       max = tab[1];
11:    }
12:    else{
13:       min = tab[1];
14:       max = tab[0];
16:    }
17:    for(i=2; i<n-1; i+=2){
18:       if (tab[i+1]>tab[i]){
19:          if (tab[i+1]>max)
                max=tab[i+1];
20:          if (tab[i] <min)
                min=tab[i];
21:    }
22:    else{
23:      if (tab[i] >max) max=tab[i];
24:      if(tab[i+1]<min)min=tab[i+1];
25:    }
26:    }
27:    if ((n%2)!=0){
28:       if (tab[n] > max) max=tab[n];
29:       if (tab[n] < min) min=tab[n];
30:    }
31:    return 0;}
```

Q1) Modify both programs until the number of iterations **(Nbr_It)** in each program has been counted and displayed.

Q2) Fill in the table **(Nbr_It)** by running the two programs with different values of ***n***.

N		**10**	**100**	**1000**	**10000**	**100000**	**1000000**	**10000000**
Nbr_It	**P1**							
	P2							

Q3) Modify both programs so as to count and display the number of elementary operations **(Nbr_Op)** (assignment, comparison, addition, modulo and access to an array element) in each of the pieces of code **P1** (line 8 to 13) and **P2** (line 8 to 30).

Q4) Fill in the table **(Nbr_Op)** by running the two programs with different values of ***n***.

N		**10**	**100**	**1000**	**10000**	**100000**	**1000000**	**10000000**
Nbr_Op	**P1**							
	P2							

Q5) Give the theoretical number of operations for both pieces of code **P1** (line 8 to 13) and **P2** (line 8 to 30).

Q6) Compare the execution times (runtimes) of the two pieces of code **P1** (line 8 to 13) and **P2** (line 8 to 30). (e.g., using the clock function).

Example of runtime calculation

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

```
int main()
        {
                clock_t T1, T2; double time;

                T1 = clock();           // take the clock before the block
                { Instruction block }
                T2 = clock();           // takes clock after function call

                // calculates execution time in seconds
                time = (double)(T2 - T1) / (double)CLOCKS_PER_SEC;
                cout << "execution time is: " << time<< "second(s) "<<endl;
        }
```

Q7) What do you observe? What do you conclude?

Exercise I.3 (Iterative vs Recursive)

The Fibonacci sequence is defined as follows:

$$F(n) = F(n-1) + F(n-2) \; for \; n \geq 2 \; with \; F(0) = F(1) = 1$$

Consider the following two functions **Fib1** and **Fib2**, which calculate the **n**th term of the Fibonacci sequence defined below:

Iterative version

```
int Fib1 (int n){
     int F0=1,F1=1,Fn;
     if (n <= 1)
        return 1;
     else
     {
        for (int i = 2; i <= n; i++){
            Fn= F1+F0;
            F0=F1;
            F1=Fn;
        }
     return Fn;
}
```

Recursive version

```
int Fib2 (int n)

{

    if (n <= 1)

       return 1;

    else

       return Fib2(n-1) +Fib2(n-2);

}
```

Q1) Compare the execution times (runtimes) of these two functions for different values of n, filling in the following table.

N		**10**	**20**	**30**	**40**	**50**	**10000**	**10000000**
Runtimes	**Fib1**							
	Fib2							

Q2) Based on the execution times obtained, which function would you choose to calculate the **n**th term of the Fibonacci sequence? Justify your answer.