ANÁLISIS Y DISEÑO DE ALGORITMOS I
Práctico Nº 1. Análisis de eficiencia de algoritmos
Segunda Parte

1. Determine el tiempo de ejecución en notación Big-Oh de los siguientes fragmentos de programas C++:

a.
```
void Calculo (double a, double b, double c) {
   double resultado;
   resultado = a + b + b*c + (a+b-c)/(a+b) + 4.0;
   cout << resultado << endl;
}
```

b.
```
float Suma (float arreglo[ ], int cantidad) {
      float suma= 0;
      for (int i = 0; i < cantidad; i++)
            suma += arreglo [i];
      return suma;
   }
```

c.
```
unsigned int Fibonacci (unsigned int i) {
      int Fi_1 = 1, Fi_2 = 1, Fi= 1;
      for (int j = 2; j <= i; j++){
            Fi= Fi_1 + Fi_2;
            Fi_2 = Fi_1;
            Fi_1 = Fi;
      }
      return Fi;
   }
```

d.
```
void Transpuesta (int Matriz [][SIZE]){
      for (int i = 0; i < SIZE; i++)
            for (int j = i+1; j< SIZE; j++){
               int aux= Matriz [i][j];
               Matriz [i][j]= Matriz [j][i];
                  Matriz [j][i]= aux;
            }
   }
```

e.
```
void productoMatricesCuadradas (double a[N][N], double b[N][N],
                                double c[N][N]){
      for (int i= 0; i < N; i++)
         for (int j= 0; j < N; j++){
            c[i][j]= 0.0;
            for (int k= 0; k < N ; k++)
                c[i][j]+= a[i][k] * b[k][j];
         }
   }
```

f.
```
void CaminosMinimos (float costos[][SIZE],float A[][SIZE]) {
   for (int i = 0; i < SIZE; i++)
        for (int j = 0; j < SIZE; j++)
             A[i] [j] = costos [i] [j];
     for (int k = 0; k < SIZE; k++)
          for (int i = 0; i < SIZE; i++)
             for (int j = 0; j < SIZE; j++)
                  if (A[i][j] > A[i][k] + A[k][j])
                          A[i][j] = A[i][k] + A[k][j];
   }
```

g.
```
#include <iostream>
   using namespace std;
   const int m = 100000000;
   const int m1 = 10000;
   const int b = 31415821;
   int a;

   int mult (int p, int q){
      int p1, p0, q1, q0;
      p1 = p/m1; p0 = p%m1; // el operador binario "%" es el módulo,
                            //%calcula el resto de la división entera
      q1 = q/m1; q0 = q%m1;
      return (((p0 * q1 + p1 * q0) % m1) * m1 + p0 * q0) % m;
   }

   int random (){
      a= (mult (a,b) + 1) % m;
      return a;
   }

   void random_1 (){
        cin >> a;
        for (int i= 1; i <= 1000000; i++)
             cout << random() << endl;
   }
   void random_2 (){
        int i, N;
        cin >> N >> a;
        for (i= 1; i <= N; i++)
             cout << random() << endl;
   }

   int main (){
        random_1();
        random_2();
        return 0;
   }

//----------fin ejercicio g -----------------
```

h.
```
int BusquedaBinariaIterativa( int Numeros[], int inicio,
                              int fin, int numeroBuscado ) {
   // Números es un arreglo de enteros ordenados de menor a mayor
      while (inicio <= fin) {
            int pos = (inicio + fin) / 2;
            if (numeroBuscado < Numeros[pos])
                  fin = pos - 1;
            else
                  if (numeroBuscado > Numeros[pos])
                        inicio = pos + 1;
                  else
                        return pos;
      }
      return -1;
   }
```

i.
```
int Contar (int n, int m){
      int j, contadorSentencias = 0, i = 1;
      while (i <= m) {
            j = n;
            while (j != 0){
                  j = j / 2; contadorSentencias ++;
            }
            i++;
      }
      return contadorSentencias;
   }
```

j.
```
void par_impar (int k)  {
      int i, j, contadorSentencias = 0;
      for (i= 1; i <= k; i++) {
            for (j= 1; j <= i; j++)
                  contadorSentencias++;
            if (i % 2 == 0)
                  for (j= i; j <= k; j++)
                        contadorSentencias++;
      }
   }
```

k.
```
void Contador (int n) {
      int x, m, cuenta;
      for (m = 2; m <= n; m++) {
            cuenta = 0;
            x = 2;
            while (x <= m) {
                  x = 2*x;
                  cuenta++;
            }
            cout << cuenta;
      }
   }
```

2. Codifique en C++ los problemas que se listan a continuación y determine la complejidad temporal en notación Big-Oh:

a. Dado un arreglo de números enteros, escribir una función en C++ que verifique si un valor determinado pertenece o no al arreglo.

b. Escribir una función que reciba como parámetro un arreglo de N números naturales, busque el elemento "mayoría" y retorne si existe el elemento mayoría y, en caso positivo, la cantidad de veces que aparece en el arreglo. El elemento mayoría es aquel que aparece más de N/2 veces en el arreglo.

c. Escribir en C++ una función que calcule la potencia de un número entero, sin utilizar funciones de bibliotecas.

```
int Potencia(int base, int exponente)
{
// escriba aquí su código
}
```

d. Dado un arreglo de enteros implementar en C++ las siguientes funciones:

```
void Ordenamiento_Seleccion (double arreglo [], int n)
{
// escriba aquí su código
}

void Ordenamiento_Burbuja (double arreglo [], int n)
{
// escriba aquí su código
}
```