UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
ALGORITMOS Y ESTRUCTURAS DE DATOS

PRÁCTICA DE COMPLEJIDAD EN TIEMPO Y ESPACIO

1. Explique el concepto de dominancia de funciones (gráfica y formalmente). Explique cuál es su fin, es decir, ¿para qué es utilizada?
2. Demuestre formalmente las siguientes propiedades del concepto de dominancia de funciones:
   - Si $f(n) = O(g(n))$ y $g(n) = O(h(n))$ entonces $f(n) = O(h(n))$
   - $f(n) = O(f(n))$
   - Si $f(n) = O(g(n))$ y $g(n) = O(f(n))$ entonces $f(n) = g(n)$
3. ¿Qué entiende usted por el término complejidad en tiempo?
4. Si dos algoritmos tienen el mismo orden teórico en tiempo de ejecución, entonces ¿ambos tardarán el mismo tiempo en ejecutarse en la práctica? Suponga que ambos se ejecutan en plataformas totalmente iguales. Explique su respuesta
5. Discuta la veracidad de la siguiente afirmación: $n^2 = O(n\log_2 n)$. En caso de ser cierto, demuéstrelo formalmente. En caso de ser falso, indique dos formas distintas que permitan concluir que dicho razonamiento es erróneo.
6. Para las siguientes f(n), halle una cota g(n), y demuestre mediante el concepto de dominancia de funciones que $f(n) = O(g(n))$
   a) $f(n) = 8n^3 – 576n^2 + 832n – 248$
   b) $f(n) = \ln(an) + b$
   c) $f(n) = n^2 – \log_2 n + 7$
   d) $f(n) = an^2 + bn + c$
   e) $f(n) = n + n^2 + n^3$
   f) $f(n) = a2^{bn} + c$
   g) $f(n) = 3^x + \log_2(n + 1)$
7. ¿Es cierto que el siguiente código:

```
if (Cond) {
   X = X + 1;
} else {
   X = X – 1;
}
```

   es de O(1)? (justifique en detalle su respuesta)
8. ¿Bajo qué condiciones es un algoritmo de O(1)?
9. Supongamos que el tiempo de ejecución de un procedimiento A es una constante M. Calcular el orden de magnitud de la función complejidad T(n) que mide el tiempo de ejecución de los siguientes algoritmos, donde n es el tamaño de la entrada (N en los algoritmos), y demuestre formalmente mediante el concepto de dominancia de funciones cual es su complejidad.
   a) Algoritmo 1:

```
int I = 1;
do {
  int J = 1;
    do {
      int K = 1;
      do {
        A();
        K = K + 1;
      } while (K != J);
      J = J + 1;
    } while (J != I);
    I = I + 1;
} while (I != N);
```

Algoritmo 2:

```
int J = N;
while (J > 1) {
  A();
  J = J/2;
}
```

10. Analice el tiempo de ejecución T(n) de los siguientes fragmentos de programa y calcule su orden de complejidad:

```
int sum = 0;
for (int i = 0; i < n; i++) {
  for (int j = 0; j < n*n; j++) {
    sum = sum + 1;
  }
}
```

```
int i = 1;
int x = 0;
while (i <= n) {
  x = x + 3;
  i = i * 2;
}
```

```
int x = 0;
for (int i = 0; i < n; n++) {
  for (int j = 0; j < i*i; i++) {
    for (int k = 0; k < j; j++) {
      x = x + 2;
    }
  }
}
```

```
int sum = 0;
for (int i = 0; i < n; i++) {
  for (int j = 0; j < i; i++) {
    sum = sum + 1;
  }
}
```

```
int i = 1;
int x = 0;
do {
  int j = 1;
  while (j <= n) {
    x = x + 1;
    j = j * 2;
  }
  i = i + 1;
}while (i==n);
```

11. Suponga que cuenta con un arreglo desordenado de enteros. Se desea implementar una operación selectora, que dado un parámetro k, retorna el entero correspondiente a la posición k del arreglo si este estuviera ordenado (obviamente, sin ordenarlo). Defina un algoritmo que sea eficiente y determinístico, es decir, dado cierto k y cualquier arreglo, el tiempo de búsqueda sea el mismo. Calcule formalmente la complejidad en tiempo del programa.

12. Calcule la complejidad en tiempo del siguiente algoritmo:

```
void  comp (int n) {
  for (int i = 0; i < n – 1; i++) {
    int j = 0;
    while (i < n) {
      j = j + 1;
      i =  i + 1;
    }
    if (j < n) {
      for (int k = 1; k < n; k++) {
        for (int k2 = 1; k < n; k++) {
          i = k*k2;
        }
      }
    }
  }
}
```

13. Dado el siguiente algoritmo:

```
void  alfa ( ) {
  int N,X,Y,Z,M;
  X=0; Y=0; Z=0;
  cin >> N;
  M = pow(2, N);
  for (int I = 1; I < M; i++) {
    Z = (4*I) + 8 + Z;
    Y = Y + (2 * I);
    X = X + 1;
  }
}
```

a) Calcule la función T(n) del algoritmo e indique su orden de complejidad en tiempo.
b) Reescriba este algoritmo para que su orden de complejidad sea de O(1). Recuerde que el nuevo algoritmo debe tener los mismos resultados que el original.

14. Para el siguiente algoritmo indique su complejidad en tiempo. Explique detalladamente que ocurre.

```
for (int i = 1; i< n; i++) {
 int j,k1,k2;
  for (j = 1; j < n-1; j++) {
    i = i + 1;
    j = j + 1;
  }
  for (k1 = i; k < n; k++) {
    for (k2 = k1; k2 < n; k++) {
      k1 = k1 +1;
      k2 = k2 + 1;
    }
  }
}
```

15. Calcule la complejidad del siguiente algoritmo:

```
int j = 1;
while (j < N) {
  j = j * b;
}
```

Donde b es una constante, dado que:

a) N también es un valor constante
b) N es la entrada de datos del programa

16. Calcule la complejidad en tiempo para los siguientes fragmentos de código:

- ```
  int I = 1;
  int X = 0;
  while (I <= n) {
    x++;
    I += 3;
  }
  ```
- ```
  int I = 1;
  int X = 0;
  while (I <= n) {
    X +=3;
    I *= 3;
  }
  ```
- ```
  for (int i =1; i <= n; i++) {
    int j = 1;
    while (j <= i ) {
      j *= 2;
      x += 2;
    }
  }
  ```
- ```
  int X = 0;
  for (i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
      for (int k = 1; k <= n; k++) {
        X += 2;
      }
    }
  }
  ```
- ```
  int X = 0;
  for (int i = 1; i <= n; i++) {
    for(int j = 1; j <= i; j++) {
      for(int k = 1; k <= j ; k++) {
        X+=2;
      }
    }
  }
  ```

17. Para el festival OTI de este año se diseño la siguiente estructura:

```
struct Cantante {
  char nombre [512];
  char paisOrig[10];
  char nomCanc;
  char autor;
  int puntosObt;
  bool sexo;
  bool esInvitado;
};

Cantante festival [20];
```
```

- Calcule la cantidad de memoria requerida para la estructura del objeto festival.
- Realice un algoritmo que indique el número de invitados que se encontraban en el festival (asuma que el arreglo es de tamaño N). Calcule la complejidad en tiempo del mismo.
- Realice un algoritmo que permita saber los datos del participante que haya obtenido el mayor puntaje. Calcule la complejidad en tiempo del mismo.

18. Para el siguiente bloque de programa:

```
struct Vivienda {
  float Costo;
  char Direccion[1024];
  int Pisos;
  bool TieneVigilancia;
  int Metros;
  int Altura;
};
Vivienda Urb [15];
```

- Calcule la cantidad de memoria requerida para la estructura del objeto Urb.

19. Para el siguiente bloque de programa:

```
struct Fecha {
  int Dia;
  char Mes[4];
  int Año;
};
struct EntradaDeDirectorio {
  char Nombre [256];
  Fecha Creacion;
  int NumeroDeArchivos;
  bool EstaCompartido;
};
```

- Calcule la cantidad de memoria requerida para la estructura del objeto EntradaDeDirectorio

20. Dada la siguiente estructura de datos:

```
struct XG {
  int Elemento [10] [1 .. 10];
};
struct YG {
  int x, y;
  XG Milista;
};
struct K {
  YG x;
  XG y;
};
K E [20];
```

- Calcule la complejidad en espacio de la variable E.

21. Considere las siguientes definiciones de tipo:

```
struct Persona {
  char Nombre [50];
  bool EdoCivil;
  bool Sexo;
  int Edad;
  char DirTrab [512];
  int CI_Rif;
  bool PersonaJuridica;
};
Persona JuntaCondominio [10];
```

```
struct Edificio {
  char Nombre [50];
  int CantApartamentos;
  JuntaCondominio Junta;
  Char Dirección [100];
};
Edificio Avenida [25];
```

- Indique el costo en memoria de la variable Avenida;