Parcial - 1C2024 - Algo2Mendez

1

1.a

Calcule la complejidad computacional los siguientes algoritmos. Justifique cómo llegó a la respuesta. Si tiene que tomar alguna suposición para llegar al resultado, explíque cuál es la suposición. Si no es posible encontrar una solución cerrada, explíque por qué y proponga un resultado alternativo que acote el problema. Justifique cada decisión.

```
void f1(pila_t* pila)
{
    while (!pila_vacia(pila))
        pila_quitar(pila);
}
```

```
int f2(int n)
{
    int k=0;

    for (int i=0; i<n; i++) {
        for (int j=0;j<n;j++) {
            if (j==10)
                i++;
            k++;
        }
    }

    return k;
}
```

1.b

Calcule la complejidad computacional los siguientes algoritmos. Justifique cómo llegó a la respuesta. Si tiene que tomar alguna suposición para llegar al resultado, explíque cuál es la suposición. Si no es posible encontrar una solución cerrada, explíque por qué y proponga un resultado alternativo que acote el problema. Justifique cada decisión.

```
void f1(pila_t* pila)
{
    while (!pila_vacia(pila))
        pila_quitar(pila);
}
```

```c
int f2(int n)
{
    int k=0;

    for (int i=0; i<n; i++)
        for (int j=0;j<n/2;j++)
            k++;

    return k;
}
```

2

2.a

Explique cómo funciona Quicksort y aplique el algoritmo al vector para ordenarlo de menor a mayor. Muestre el procedimiento paso a paso y justifique la complejidad computacional. Para este ejercicio se pide que el pivote sea el elemento central del vector.

```
V = [2, 3, 5, 0, 2, 1, 9, 7];
```

2.b

Explique cómo funciona Quicksort y aplique el algoritmo al vector para ordenarlo de mayor a menor. Muestre el procedimiento paso a paso y justifique la complejidad computacional. Para este ejercicio se pide que el pivote sea el elemento central del vector.

```
V = [2, 3, 5, 0, 2, 1, 9, 7];
```

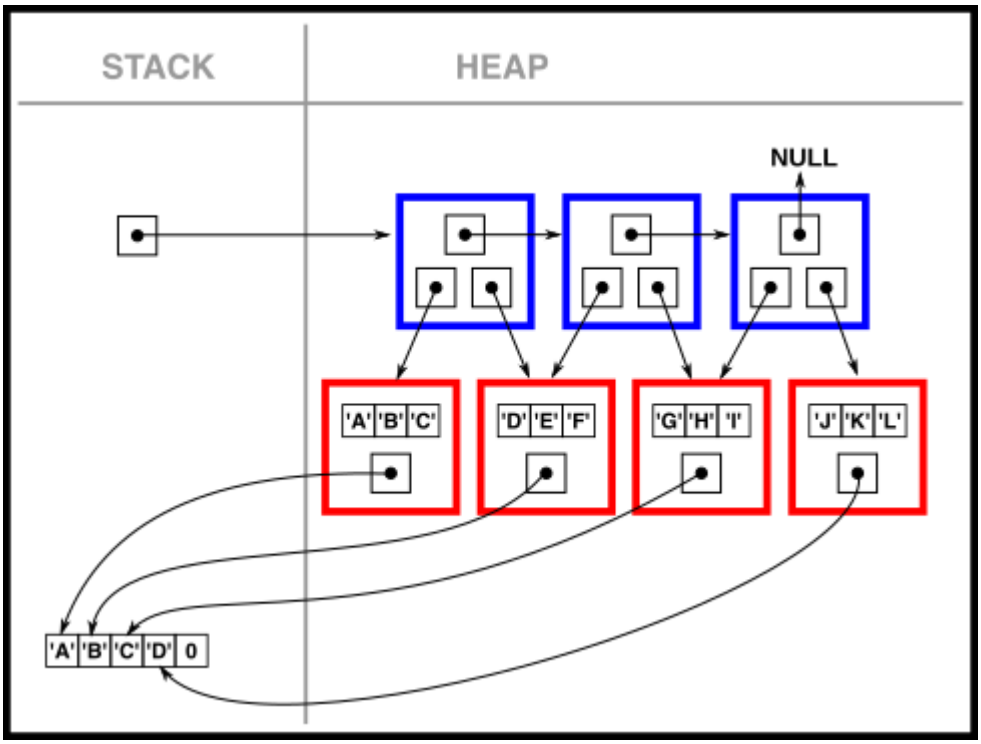
3

3.a

Escriba un programa en C que permita armar en memoria las estructuras presentadas en la imagen. Se pueden utilizar variables auxiliares extra en el stack si es necesario.

Agregue también el código para **liberar toda la memoria de forma correcta**. Notese que en el esquema las estructuras se representan con recuardos de colores y los punteros con cuadrados con un punto en el centro. Por otro lado recuerde que los elementos que se encuentran pegados físicamente (horizontal o vertical) representan datos contiguos en memoria, mientras que los elementos separados no.

ATENCION: Este ejercicio evalúa el uso de punteros y memoria dinámica. **Es fundamental que los accesos a memoria sean correctos para que el ejercicio esté aprobado**.

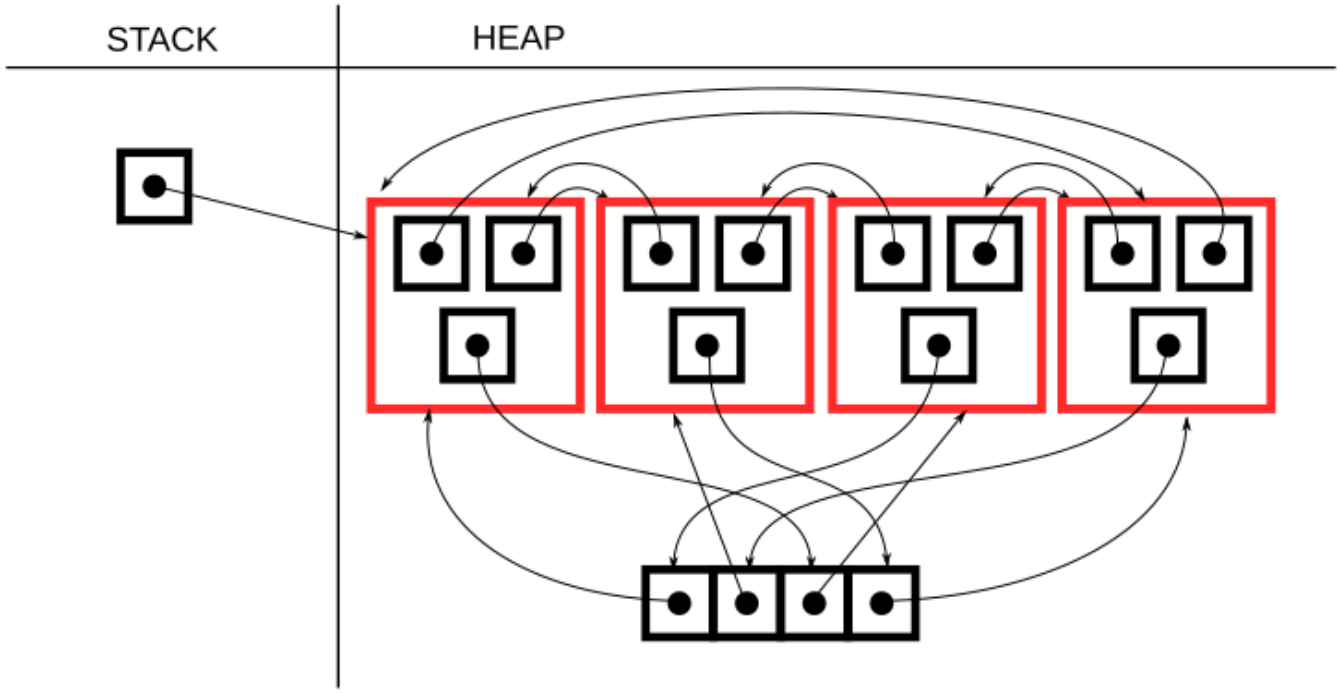


3.b

Escriba un programa en C que permita armar en memoria las estructuras presentadas en la imagen. Se pueden utilizar variables auxiliares extra en el stack si es necesario.

Agregue también el código para **liberar toda la memoria de forma correcta**. Notese que en el esquema las estructuras se representan con recuardos de colores y los punteros con cuadrados con un punto en el centro. Por otro lado recuerde que los elementos que se encuentran pegados físicamente (horizontal o vertical) representan datos contiguos en memoria, mientras que los elementos separados no.

ATENCION: Este ejercicio evalúa el uso de punteros y memoria dinámica. **Es fundamental que los accesos a memoria sean correctos para que el ejercicio esté aprobado**.



4

4.a

Dado el siguiente código

```
void mostrar_abb(abb_t* abb)
{

    puts("INORDEN: ");
    abb_con_cada_elemento(abb, INORDEN, mostrar_elemento, NULL);

    puts("\nPREORDEN: ");
    abb_con_cada_elemento(abb, PREORDEN, mostrar_elemento, NULL);

    puts("\nPOSTORDEN: ");
    abb_con_cada_elemento(abb, POSTORDEN, mostrar_elemento, NULL);
}
```

Y la siguiente salida por pantalla:

```
INORDEN: 1 8 6 0 9 18 13 25
PREORDEN: 0 1 6 8 9 13 18 25
```

Muestre cuál es la línea que falta mostrar por pantalla. Explique y justifique el razonamiento para llegar a dicho resultado.

4.a

Dado el siguiente código

```
void mostrar_abb(abb_t* abb)
{

    puts("INORDEN: ");
    abb_con_cada_elemento(abb, INORDEN, mostrar_elemento, NULL);

    puts("\nPREORDEN: ");
    abb_con_cada_elemento(abb, PREORDEN, mostrar_elemento, NULL);

    puts("\nPOSTORDEN: ");
    abb_con_cada_elemento(abb, POSTORDEN, mostrar_elemento, NULL);
}
```

Y la siguiente salida por pantalla:

```
INORDEN: 3 2 5 4 6 7 1 8
PREORDEN: 1 2 3 4 5 6 7 8
```

Muestre cuál es la línea que falta mostrar por pantalla. Explique y justifique el razonamiento para llegar a dicho resultado.

5

Escriba un algoritmo bubble sort recursivo (sin utilizar **for**, **while**, etc) para un vector de enteros.