

Estructuras de Datos y Algoritmos I

segundo examen parcial 2021-2

Nombre: Sánchez Alvirde Andrés Iván

1. Escriba en el paréntesis la letra que corresponde al concepto:

- (E) Estructura de datos lineal, en la cual el elemento obtenido a través de la operación ELIMINAR está predefinido y es el que se encuentra al inicio de la estructura
- (C) Conjunto de nodos alineados de manera lineal (uno después de otro) y unidos entre sí por una referencia
- (D) Está constituida por un conjunto de nodos alineados de manera lineal (uno después de otro) y unidos entre sí por dos referencias,
- (B) Unidad básica de una lista
- (A) Estructura de datos lineal y dinámica, en la cual el elemento obtenido a través de la operación ELIMINAR está predefinido, debido a que implementa la política Last-In, First-Out (LIFO)

- A. Pila
- B. nodo
- C. Lista simple
- D. Lista doble
- E. Cola

2. ¿En que consiste el método de burbuja?

Consiste en comprar todos los elementos de una lista, si se cumple que uno es mayor o menor que otro, entonces los intercambia de posición.

3. ¿Cuál es la diferencia entre el método de burbuja y el quick sort?

El de burbuja va recorriendo varias veces el arreglo comparando numero por numero cambiando de posición a los que son mayor o menor que, mientras que el quick sort usa una técnica conocida como “divide y vencerás” en la cual se divide en dos el arreglo que va a ser ordenado y se llama recursivamente para ordenar las divisiones, hace uso de un pivote el cual va ordenando la posición de acuerdo a ese pivote.

4. ¿Qué es y para que sirve la recursividad?

La recursividad es una función la cual dentro de una línea de código de esa misma función se llama así misma.

El propósito de la recursividad es dividir un problema en problemas más pequeños y esto para que de una manera la solución del problema se vuelva ligero.

5. ¿Cuál es la diferencia entre la notación asintótica y la notación omega?

La notación asintótica se trata sobre el peor de los casos a la hora de ejecutar un programa debido a que este se tarda mucho tiempo.

En cuanto a la notación omega será la que nos muestra el estar en medio de dos notaciones la cual son la O y la theta por lo que se puede ver que no es peor o mejor de los casos, simplemente esta en un intermedio de ambas.

Instrucciones:

La teoría la pueden contestar en el mismo examen

Los programas deberán poner código y captura de ejecución

Todo en un mismo pdf

6. Elaborar un programa donde se pida un numero al usuario si es par almacenarlo en una pila si es impar almacenarlo en una cola, en C

CÓDIGO	EJECUCIÓN
<pre>#include <stdio.h> #include <stdlib.h> struct nodo{ int dato; nodo *siguiente; }; nodo *fin=NULL; nodo *pila; nodo *inicio=NULL; void pushp(); void mostracol(); void mostrpil(); main(){ int op, op1; do{ printf("%cQu%c operacion deseas realizar?\n", 168,130); printf("1.- Insertar dato\n"); printf("2.- Mostrar datos cola\n"); printf("3.- Mostrar datos pila\n"); printf("4.- Mostrar datos de ambas\n"); printf("5.- Salir\n"); scanf("%d",&op); switch(op){ case 1: pushp(); break; case 2:mostracol(); system ("pause"); system ("cls"); break; case 3:mostrpil(); system ("pause"); system ("cls"); break; case 4: printf("\nDatos pila:\n"); mostrpil(); printf("\n"); printf("\nDatos cola:\n"); mostracol(); system ("pause");</pre>	

```

        system ("cls");
        break;
        case 5:
            printf("\n Hasta luego");
            break;
        default:
            printf("opci%cn, no es
valida\n", 162);
            break;
    }
}while(op<5);
}

void mostrpil(){
    nodo *indice = pila;
    if(indice==NULL){
        printf("La pila esta vacia");
    }
    else{
        while(indice!=NULL){
            printf("[%d]\t",indice-
>dato);
            indice = indice->siguiente;
        }
        printf("\n");
    }
}

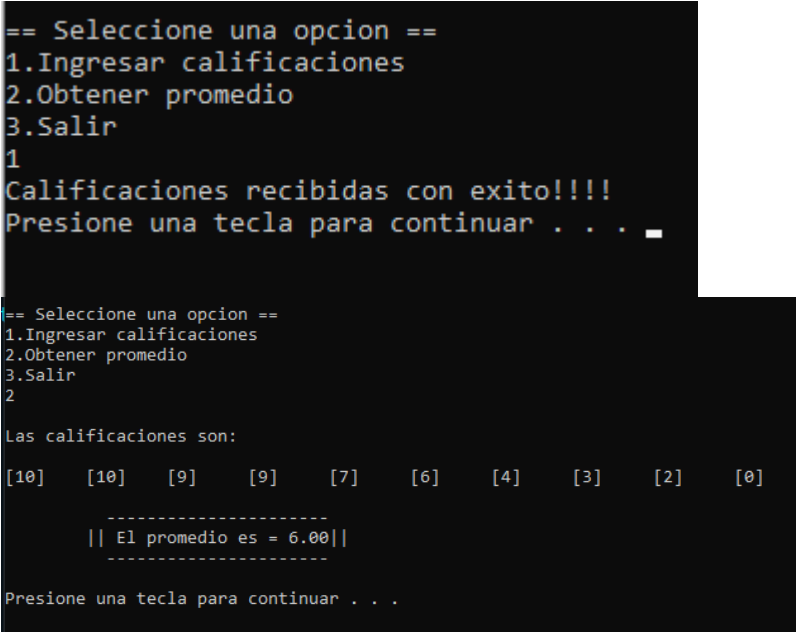
void mostrcol(){
    nodo *indice=inicio;
    if(indice==NULL){
        printf("La cola esta vacia");
    }
    else{
        while(indice!=NULL){
            printf("[%d]\t",indice-
>dato);
            indice=indice->siguiente;
        }
        printf("\n");
    }
}

void pushp(){
    int x,n;
    printf("Escriba el numero de datos que va a
ingresar\n");
    scanf("%d",&n);
    for (int i=1;i<=n;i++){
        printf("\nIngrese el dato %d:\n",i);
        scanf("%d",&x);
        if(x%2==0 || x==0){
            nodo
*nuevo=(nodo*)malloc(sizeof(nodo));
            nuevo->siguiente=pila;
            nuevo->dato=x;

```

<pre> pila = nuevo; } else{ nodo *nuevo = (nodo*)malloc(sizeof(nodo)); nuevo->dato = x; nuevo->siguiente = NULL; if(inicio==NULL){ inicio=nuevo; fin=nuevo; } else{ fin->siguiente=nuevo; fin=nuevo; } } } system ("pause"); system ("cls"); } </pre>	
--	--

7. Elaborar un programa que obtenga el promedio de 10 calificaciones obtenidas de manera aleatoria entre 0 y 10, deberá mostrar las calificaciones ordenadas de mayor a menor, en C

CÓDIGO	EJECUCIÓN
<pre> // #include <stdio.h> #include <stdlib.h> #include <time.h> void qs(int arr[],int i,int j); float suma=0.0; int n=10; int main(){ int i,j; int op,b=0; do{ printf("\n== Seleccione una opcion == "); printf("\n1.Ingresar calificaciones"); printf("\n2.Obtener promedio"); printf("\n3.Salir\n"); scanf("%d",&op); switch(op){ case 1: int *arr; arr=(int *)malloc(n*sizeof(int)); srand(time(NULL)); </pre>	 <pre> == Seleccione una opcion == 1.Ingresar calificaciones 2.Obtener promedio 3.Salir 1 Calificaciones recibidas con exito!!!! Presione una tecla para continuar . . . _ == Seleccione una opcion == 1.Ingresar calificaciones 2.Obtener promedio 3.Salir 2 Las calificaciones son: [10] [10] [9] [9] [7] [6] [4] [3] [2] [0] ----- El promedio es = 6.00 ----- Presione una tecla para continuar . . . </pre>

```

for
(i=0;i<n;i++){

    arr[i]=rand()%11;

}

printf("Calificaciones recibidas con
exito!!!\n\a");

system("pause");

system("cls");

break;
case 2:
    printf("\nLas
calificaciones son:\n");
    qs(arr,0,i);
    printf("\n");
    for
    (i=0;i<n;i++){

        printf("[%d]\t",arr[i]);

        suma=suma+arr[i];
    }
    float
    promedio;
    suma =
    suma/n;

    printf("\n");
    printf("\n");
    printf("\t ---
-----");

    printf("\n\t | El promedio es =
%.2f | \n\a",suma);

    printf("\t ---
-----");

    printf("\n");
    printf("\n");

    system("pause");

    system("cls");

    break;
case 3:
    b=1;
    break;

}
}while(b==0);
}
void qs(int arr[],int i,int j){

    int inicio=i,fin=j,pibote,temp;

```

<pre> pibote=arr[i]; //arreglo con pibote, ultimo elemento es el pibote do{ while(arr[i]>pibote) //elementos menores al pibote i=i+1; while(arr[j]<pibote) j=j-1; if(i<=j){ temp=arr[i]; arr[i]=arr[j]; arr[j]=temp; i=i+1; j=j-1; } }while(i<=j); if(inicio<j) qs(arr,inicio,j); if(i<fin) qs(arr,i,fin); } </pre>	
--	--

8. Implementar una lista doblemente ligada con las funciones, buscar, insertar, borrar y mostrar, en C

CÓDIGO	EJECUCIÓN
<pre> #include <stdio.h> #include <stdlib.h> struct nodo{ int dato; nodo *siguiente; nodo *anterior; }; nodo *primero=NULL; nodo *ultimo=NULL; void insertarNodo(); void buscarNodo(); void eliminarNodo(); void mostrarLista(); int main(){ int op; do{ printf("\n\t====="); printf("\tLISTA DOBLE LIGADA"); printf("\t====="); printf("\n\tEscoge una opcion:\n"); printf("1.-Insertar\n"); printf("2.-Buscar\n"); printf("3.-Eliminar\n"); printf("4.-Mostrar\n"); </pre>	<pre> ===== LISTA DOBLE LIGADA ===== Escoge una opcion: 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir 1 Ingresa el dato: 10 Dato ingresado correctamente Presione una tecla para continuar . . . Escoge una opcion: 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir 2 Ingresa el dato a buscar: 10 Se encuentra el dato Presione una tecla para continuar . . . Escoge una opcion: 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir 4 [10] Presione una tecla para continuar . . . </pre>

```

        printf("5.-Salir\n");
        scanf("%d",&op);
        switch(op){
            case 1:
                insertarNodo();
                system("pause");
                system("cls");
                break;
            case 2:
                buscarNodo();
                system("pause");
                system("cls");
                break;
            case 3:
                eliminarNodo();
                system("pause");
                system("cls");
                break;
            case 4:
                mostrarLista();
                system("pause");
                system("cls");
                break;
            case 5:
                printf("\nVuelva
pronto\n");
                break;
        }
    }while(op<5);
}

void insertarNodo(){
    nodo *nuevo=(nodo*)malloc(sizeof(nodo));
    printf("Ingresa el dato:\n");
    scanf("%d",&nuevo->dato);
    if(primeros==NULL){
        primeros=nuevo;
        primeros->siguiente=primeros;
//ligarse
        ultimo=primeros;
        primeros->anterior=ultimo;
    }
    else{
        ultimo->siguiente=nuevo;
        nuevo->siguiente=primeros;
        nuevo->anterior=ultimo;
        ultimo=nuevo;
        primeros->anterior=ultimo;
    }
    printf("\nDato ingresado
correctamente\n");
}

void buscarNodo(){

```

```

Escoge una opcion:
1.-Insertar
2.-Buscar
3.-Eliminar
4.-Mostrar
5.-Salir
3
Ingresa el dato a buscar:
10
Se encuentra el dato
Nodo eliminado
Presione una tecla para continuar . . . _

```

```

int nodoBuscado,encontrado=0;
nodo *actual=(nodo*)malloc(sizeof(nodo));
actual=primero;
printf("Ingresa el dato a buscar:\n");
scanf("%d",&nodoBuscado);
if(primero!=NULL){
    do{
        if(actual->
dato==nodoBuscado){
            printf("Se encuentra
el dato\n");
            encontrado=1;
        }
        actual=actual->siguiente;
    }while(encontrado==0 &&
actual!=primero);
    if(encontrado==0){
        printf("No se encuentra el
dato\n");
    }
}
else{
    printf("La lista se encuentra
vacía\n");
}
}

void eliminarNodo(){
    int nodoBuscado,encontrado=0;
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    nodo *ant=(nodo*)malloc(sizeof(nodo));
    ant=NULL;
    printf("Ingresa el dato a buscar:\n");
    scanf("%d",&nodoBuscado);
    if(primero!=NULL){
        do{
            if(actual->
dato==nodoBuscado){
                printf("Se encuentra
el dato\n");
                if(actual==primero){
                    primero=primero->siguiente;
                    primero=ultimo->siguiente;
                    ultimo=ultimo->siguiente;
                }
                else{
                    if(actual==ultimo){
                        ultimo=ant;
                    }
                }
            }
        }while(actual!=ultimo);
    }
}

```


<pre> primero- >anterior=ultimo; } else{ ant- >siguiente=actual->siguiente; actual- >siguiente->anterior=ant; } printf("Nodo eliminado\n"); encontrado=1; } ant=actual; actual=actual->siguiente; }while(encontrado==0 && actual!=primero); if(encontrado==0){ printf("No se encuentra el dato\n"); } else{ free(ant); } } else{ printf("La lista se encuentra vacía\n"); } } void mostrarLista(){ nodo *actual=(nodo*)malloc(sizeof(nodo)); actual=primero; if(actual!=NULL){ do{ printf("[%d]\t",actual->dato); actual=actual->siguiente; }while(actual!=primero); printf("\n"); } else{ printf("La lista esta vacía\n"); } } </pre>	
---	--

9. Determinar si en una cadena existen las 5 vocales en orden, las vocales se pueden repetir, puede haber entre ellas consonantes o no, se deberá usar funciones recursivas, en C

CÓDIGO	EJECUCIÓN

10. Obtener el factorial de un número usando recursividad, en C

CÓDIGO	EJECUCIÓN
<pre>#include<stdio.h> int factorialNumero(int n) { if (n > 1) { return n * factorialNumero(n-1); } else { return 1; } } int main() { int j; printf("Recursividad para obtener el Factorial.\n\n"); printf("Ingrese el numero al cual se le obtendra su factorial:\n"); scanf("%d",&j); int resultado = factorialNumero(j); printf("\nEl resultado es: %d \n\n", resultado); }</pre>	<pre>Recursividad para obtener el Factorial. Ingrese el numero al cual se le obtendra su factorial: 8 El resultado es: 40320</pre>