Pregunta 1

#include **<stdio.h>**#include **<stdlib.h>***---------------------------//implementacion de la pila-------------------------------***typedef struct** info{  
 **int** valor;  
}Info;  
  
**typedef struct** nodo{  
 Info \*elem;  
 **struct** nodo \*sig;  
}Nodo;  
  
**typedef struct** Pila{  
 Nodo \*tope;  
 **int** tamano;  
}Pila;

Diseñe las siguientes funciones para una Pila.

* Crear - Se crea la pila vacía.

Pila \*crearPila(){  
 Pila \*p;  
 **if** (p = (Pila \*)malloc(**sizeof**(Pila))){  
 p -> tope = **NULL**;  
 p -> tamano = 0;  
 }**else**{ printf(**"memoria de la pila no se pudo asignar"**);}  
 **return** p;}

* Tamaño- Regresa el número de elementos de la pila.

**int** tamanoPila(Pila \*p){  
 **return** p->tamano;  
}

* Apiliar - Se añade un elemento a la pila.

Nodo \*apilar(Pila \*p, **int** pval){  
 Info \*pinfo = crearInfo(pval);  
 Nodo \*newNodo;  
 **if** (newNodo = (Nodo\*)malloc(**sizeof**(Nodo))){  
 newNodo -> elem =pinfo;  
 newNodo ->sig = p ->tope;  
 p ->tope = newNodo;  
 p->tamano++;  
 }**else**{printf(**"memoria de nodo no pudo ser asignada"**);}  
 free(newNodo);  
}

* Se crea la información del nodo

Info \*crearInfo(int pval){

Info \*newInfo;

If(newInfo = (Info\*)malloc(sizeof(Info))){

NewInfo -> valor = pval;

}else{printf(“la asignación de memoria no se realizó”);}

return newInfo;

}

* Desapilar - Se elimina el elemento frontal de la pila.

Info \* desapilar(Pila \*p){  
 Nodo \*aux;  
 Info \*inf;  
 **if** (esVacia(p)){  
 printf(**"la pila esta vacia no se puede desaplar"**);  
 }**else**{  
 aux = p ->;  
 inf = aux ->elem;  
 p->tamano--;  
 free(aux);  
 }  
 **return** inf;  
}

*//-------------------- verificar si la pila esta vacia*

**bool** esVacia (Pila \*p){  
 **if** (p->tope = **NULL**)**return true  
 else** { **return false**;}  
}

* Tope - Devuelve el elemento que está en la cima de la pila

Nodo \*nodoTope(Pila \*p){  
 **return** p->tope;  
}

* Vacia - Devuelve cierto si la pila esta vacia o falso en caso contrario

**Pila \***esVacia (Pila \*p){  
 **if** (p->tope = **NULL**)**return true  
 else** { **return false**;}}

* Llena - Devuelve cierto si la pila esta llena o falso en caso contrario

**bool** pilaLLena(Pila \*p) {  
  
 **if** (p == tamanoPila) {  
 **return true**;  
 } **else** { **return false**; }  
}

Pregunta 2

#include **<stdio.h>**#include **<stdlib.h>** *//------------------- implementación---------------------------*

**typedef struct** info{  
 **int** elemen;  
}Info;

**typedef struct** nodo{  
 Info \*elemen;  
 **struct** nodo \*sig;  
}Nodo;

**typedef struct** cola{  
 Nodo \*ini;  
 Nodo \*fin;  
 **int** tamano;  
}Cola;

* Crear - Se crea la cola vacía

Cola \*crearCola(){  
 Cola \*tre;  
 **if** (tre = (Cola\*)malloc(**sizeof**(Cola))){  
 tre -> ini = tre->fin = **NULL**;  
 tre ->tamano =0;  
 }**else**{printf(**"no se pudo asignar memoria"**);}  
 **return** tre;  
}

Info \*crearInfo(**int** pinfo) {  
 Info\*newInfo;  
 **if**(newInfo = (Info \*)malloc(**sizeof**(Info))){  
 newInfo->elemen =pinfo;  
 }**else**{printf(**"no se pudo asignar memoria"**);}  
 **return** newInfo;  
}

* Tamaño - Regresa el numero de elementos de la cola.

**int** tamCola(Cola \*tam){  
 **if** (tam ->tamano!=**NULL**){  
 printf(**"tamaño de la cola es: %d"**,tam->tamano);}  
 **else**{printf(**"la cola esta vacia"**);}  
 **return** tam->tamano;  
}

* Encolar - Se añade un elemento a la cola.

**void** encolar(Cola \*ptam,**int** pval){  
 Info \*mac = crearInfo(pval);  
 Nodo \*newNodo;  
 **if**(newNodo =(Nodo\*)malloc(**sizeof**(Nodo))){  
 **if**(estaVacia(ptam)){  
 ptam ->ini =newNodo;  
 } **else**{ptam->fin->sig=newNodo;}  
 ptam->fin=newNodo;  
 newNodo ->sig =**NULL**;  
 newNodo ->elemen =mac;  
 ptam ->tamano++;  
 }**else**{printf(**"no se pudo asignar memoria"**);}  
 free(newNodo);  
}

* Vacía - Devuelve cierto si la cola está vacía o falso en caso contrario.

Cola \* estaVacia(Cola \*ptam){  
 **if**(ptam -> == **NULL**){ **return** 1;}**else**{ **return** 0;}  
}

* Desencolar - Se elimina un elemento de la cola.

Info \*desencola(Cola \*c){  
 Nodo \*aux;  
 Nodo \*inf;  
 **if**(estaVacia(c)){printf(**"la cola esta vacia no se puede desencolar"**);  
 }**else**{  
 aux = c->ini;  
 inf = aux ->elemen;  
 c ->ini = aux ->sig;  
 c ->tamano --;  
 free(aux);  
 }  
 **return** inf;  
}

* Comienzo - Devuelve el elemento que esta al comienzo de la cola.

Cola \*primerElemnt(Cola \*itam){  
 Nodo \*aux;  
 **if**(estaVacia(itam)){printf(**"la cola esta vacia"**);  
 }**else**{aux = itam->ini;  
 **return** aux;}  
}

* Final - Devuelve el elemento que esta al final de la cola.

Cola \*ultimElemnt(Cola \*ftam){  
 Nodo \*aux;  
 **if** (estaVacia(ftam)){printf(**"la cola esta vacia"**);  
 }**else**{aux = ftam->fin;}  
 **return** aux;  
}

* Llena - Devuelve cierto si la cola está llena o falso en caso contrario.

Cola \* estaVacia(Cola \*ptam){  
 **if**(ptam -> != **NULL**){ **return** 1;}**else**{ **return** 0;}  
}