8(OCHO)

1- Una empresa posee un conjunto se registros que corresponden a la lista de fallas de cada uno de sus productos.
Cada registro esta compuesto de los siguientes campos:
Nro de producto (int)
Tipo de falla (string 40)
Tiempo medio entre fallas (horas long)
Tiempo medio de reparación (horas float).
Este conjunto de registros se encuentra ubicado en el archivo “fallas.dat” que es de tipo binario. Se desea obtener un listado que contenga los siguientes ítems:
Nro de producto
Tipo de falla
Costo de reparación.
El listado debe estar ordenado por nro de producto en forma decreciente.
Para poder calcular el costo de la reparación se debe ingresar el valor de la hora de service.
No se conoce la cantidad de datos que posee el archivo.
Si no recuerda el manejo de archivos puede emplear las funciones cuyos prototipos están en el header “infoii.h”.
Las funciones son las siguientes:
void AbreArch(char *Nombre)
Función que abre el archivo.
Recibe el nombre del archivo.
void CierraArch(void)
Función que cierra el archivo.
int LeeArch (char * Buffer,int Tamaño)
Función que lee un registro del archivo.
Recibe
Un puntero a char que es el buffer en donde almacenara el dato leído
Un entero que es el tamaño del dato a leer
Devuelve
0 si encontró el fin de archivo, en caso contrario 1.

90%

```
///Ejercicio 1
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

struct fallas {
    int numero_producto;
    char tipo_falla[40];
    float tiempo_entre_fallas;
    float tiempo_reparacion;
};

struct nodo{
    struct fallas info;
    struct nodo *next;
};

//Decalarion de funciones
int ins_final(struct nodo **, struct fallas);
void ordenar(struct nodo **);
void listar(struct nodo *, float);
void liberar_memoria(struct nodo **);

//Inicio main
int main(void) {
    struct nodo *inicio = NULL; //Declaro que el puntero apunta al nulo
    struct fallas informacion;
    float valor_hora_service;
    FILE *fp;

    if(!(fp=fopen("fallas.dat","rb"))) { //Verifico apertura de archivo
        printf("No se puede abrir el archivo\n");
        return(1);
    }
    fread(&informacion, sizeof(informacion), 1, fp);
    while(!feof(fp)) { //Cargo la lista mientras halla fallas que leer
del archivo
        ins_final(&inicio, informacion);
        fread(&informacion, sizeof(informacion), 1, fp);
    }
    fclose(fp);
    //Imprimo en pantalla
    printf("Por favor, ingrese el costo el valor de la hora de
service:\t");
    scanf("%f", valor_hora_service);
    ordenar(&inicio);
    listar(inicio, valor_hora_service);
    liberar_memoria(&inicio);
    printf("\n\n");
    system("PAUSE");
    return 0;
}
int ins_final(struct nodo **ppio, struct fallas danio) {
    struct nodo *p, *item;
    if(!(item=(struct nodo *)malloc(sizeof(struct nodo))))
        return(1);
    item->info = danio;
    item->next=NULL;
```

```
    if(!*ppio) {
        *ppio=item;
        return(0);
    }
    for(p=*ppio; p->next; p=p->next);
    p->next=item;
    return(0);
}
void ordenar(struct nodo **ppio) {
    struct nodo *p, *max, *aux=NULL;
    if(*ppio==NULL) {
        printf("\nLista VACIA");
        printf("\n\nPresione enter p/ seguir");
        fflush(stdin);
        getchar();
        return;
    }
    p=*ppio;
    while(*ppio) {
        max=p;
        while(p) {
            if(max->info.numero_producto < p->info.numero_producto)
                max=p;
            p=p->next;
        }
        if(*ppio==max)
            {
            *ppio=max->next;
            max->next=aux;
            aux=max;
            }
        else{
            p=*ppio;
            while(p->next!=max)
                p=p->next;
            p->next=max->next;
            max->next=aux;
            aux=max;
            }
        p=*ppio;
    }
    *ppio=aux;
    printf("\nLista ORDENADA");
    printf("\n\nPresione enter p/ seguir");
    fflush(stdin);
    getchar();
}
void listar(struct nodo *p, float precio_service) {
    float costo_reparacion = 0;
    if(!p)
        printf("LISTA VACIA");
    else {
        for(;p;p=p->next) {
            costo_reparacion = p->info.tiempo_reparacion *
precio_service;
            printf("%i %40s %f\n",p->info.numero_producto, p-
>info.tipo_falla, costo_reparacion);
        }
```

```
    }
    printf("\n\nPresione enter p/ seguir");
    fflush(stdin);
    getchar();
}
void liberar_memoria(struct nodo **ppio) {
    struct nodo* p;

    for(p=*ppio; p; p=*ppio){
        *ppio=p->next;
        free(p);
    }
}
```

2.- Dada la clase Frase se pide:

```
class Frase
{
private:
  char *Dato;
  int Tamanio;
public:
  Frase();
  int CuentaLetra(char);
  bool operator ==(Frase);
  void Set(char *);
};
```

cuyo constructor y el método Set son:

```
Frase::Frase()
{
  Tamanio=255;
  Dato=new char[255];
}
void Frase::Set(char * string)
{
  if(strlen(string)>Tamanio)
  {
    cout<<"string muy largo";
    return;
  }
  strcpy(Dato,string);
  Tamanio=strlen(string);
}
```

Se pide:

1.- Realizar el método "CuentaLetra" que permita obtener la cantidad de letras que tiene la frase guardada en Dato, el método recibe la letra a buscar y devuelve la cantidad de veces que encontró dicha letra.
Su prototipo es:
`int CuentaLetra(char);`

2.- Realizar la sobrecarga del operador == de modo que devuelva verdadero si son iguales y falso en caso contrario.
`bool operator ==(Frase);`

80%

```
//// Ejercicio 2.1
int Frase::CuentaLetra(char letra) {
    int contador = 0;
    for (int i = 0; i < Tamanio; i++) {
        if (Dato[i] == letra)
            contador i++;
    }
    return contador;
}

//// Ejercicio 2.2
bool Frase::operator==(Frase frase_1, Frase frase_2) {
    if(strcmp(frase_1->Dato, frase_2->Dato) == 0)
        return true;
    else
        return false;
}
```

✓

NO HAY QUE PASAR EL ARGUMENTO

NO

ES UNA VARIABLE NO UN PUNTERO