Apellido y nombre:……..…… Legajo: ……..……

Taller de Lenguajes I – Ingeniería en Computación – UNLP 2023 - 2S - Módulo II – 1ra. Fecha

Ejercicio	Puntos
1	
2	
3	
4	
5	
6	
Total	

1. **(1.20 punto)** Dado el siguiente código

```
1.  #include <stdio.h>
2.  #define BUCLE 0
3.  #define DEBUG 1
4.  int main () {
5.      int i, x = 0;
6.      #ifdef BUCLE
7.       while (x<5000) {
8.            for (i=0; i<100; i++) {
9.                  x+= 1;
10.                 #if DEBUG
11.                       printf("X= %d\n",x);
12.                 #endif
13.           }
14.       }
15.     #endif
16.     return 0;
17. }
```

   a. Indique qué imprime.
   b. Para cada afirmación indique si es verdadera o falsa y **justifique**.
      i. Al cambiar la línea 2 por "#define BUCLE 1" se compilarán más instrucciones.
      ii. Al cambiar la línea 3 por "#define DEBUG 0" se compilarán menos instrucciones.
      iii. Al cambiar la línea 10 por "#ifndef DEBUG" se compilarán menos instrucciones.

Rta:

a) Imprime los números del 1 al 5000

b) i) FALSO porque sólo pregunta si está definida no importa su valor; ii) VERDADERO porque pregunta con #if y si vale 0 la línea 11 no se compilará, iii) VERDADERO porque DEBUG está definida.

2. **(1 punto)** Defina una macro que dados tres caracteres retorne SI cuando al menos dos de ellos correspondan a un operador de suma, resta, multiplicación o división ("+", "-", "*" o "/") y NO en caso contrario. Luego escriba un programa que utilice la macro para imprimir lo siguiente:

```
En el conjunto {0,1,+} NO hay al menos 2 operadores
En el conjunto {4,+,/} SI hay al menos 2 operadores
En el conjunto {9,8,7} NO hay al menos 2 operadores
En el conjunto {*,/,+} SI hay al menos 2 operadores
```

Nota: En cada línea las palabras SI o NO son el resultado de invocar a la macro. El resto del texto es fijo.

Rta:

```
#include <stdio.h>
#define isOper(c) (((c=='+') || (c=='-') || (c=='*') || (c=='/')) ? 1 : 0)
#define hayDos(a,b,c) (isOper(a)+isOper(b)+isOper(c)>=2) ? "SI" : "NO"
int main() {
   printf("En el conjunto {0,1,+} %s hay al menos 2 operadores\n",hayDos('0','1','+'));
   printf("En el conjunto {4,+,/} %s hay al menos 2 operadores\n",hayDos('4','+','/'));
   printf("En el conjunto {9,8,7} %s hay al menos 2 operadores\n",hayDos('9','8','7'));
   printf("En el conjunto {*,/,+} %s hay al menos 2 operadores\n",hayDos('*','/','+'));
   return 0;
}
```

3. **(1 punto)** Marque con verdadero o falso las siguientes afirmaciones:

V	a) Es posible retornar un puntero a un bloque de memoria alocada dentro de una función.
V	b) Es posible reservar memoria en tiempo de ejecución para un tipo de dato compuesto.
F	c) Suponiendo que *sizeof(int)=4*, el siguiente código imprime *"tamaño = 400 bytes"* : **int ** matriz = calloc(100, sizeof(int));** **printf("tamaño = %d bytes", sizeof(matriz));**
F	d) Si sólo se va a leer un archivo binario es lo mismo abrirlo con modo **"ab"** que con **"a+b"**..
V	e) Puede utilizarse la función *fprintf()*tanto para escribir en un archivo como para escribir en pantalla.
F	f) Si *calloc* no tiene éxito al reservar memoria, su comportamiento es indeterminado.
F	g) Si se graba la misma información en un archivo binario y en un archivo de texto, el primero siempre tendrá menor tamaño.

Rta:

a) VERDADERO. La memoria alocada dentro la función continuará estando reservada luego de que la función termine.

b) VERDADERO. La reserva de memoria en forma dinámica se realiza indicando la cantidad de bytes requeridos independientemente del tipo de datos que se vaya a alojar allí.

c) FALSO. Imprime "Tamaño = 8 bytes".

d) FALSO. Si lo que se quiere es leer, no puede utilizarse "ab" porque es un modo de apertura para escritura.

e) VERDADERO. La función fprintf() puede escribir en pantalla si el primer parámetro es "stdout" o en disco si se utiliza un FILE * inicializado con fopen().

f) FALSO. La función calloc() retorna NULL si tiene algún problema con la apertura del archivo.

g) FALSO. Un archivo binario que sólo tengaun número entero ocupará 4 bytes mientras que un archivo de texto que sólo contenga un número de 1 (un) dígito ocupará 1 byte.

4. **(1.5 puntos)** Parámetros del main.

Dado el siguiente código:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){

    if (argc > 5) {
        printf("%s %d \n", argv[1], argc / atoi(argv[4]));
        printf("%s %d \n", argv[3], argv[2][1] - argv[2][0]);
    }else{
        printf("%s %d \n", argv[1], argc);
        printf("%s %d \n", argv[3], argv[1][0] - argv[1][1]);
    }
    return 0;
}
```

Indicar que imprime si luego de compilar el programa es ejecutado de la siguiente forma:

a) `C:\>proceso.exe Modulo 02 Semestre 2 end`

b) C:\>proceso.exe edicion 2023 info UNLP

Rta:

La opción a) imprime

Modulo 3

Semestre 2

La opción b) imprime

edicion 5

info 1

5. Se dispone de un archivo binario con información de alumnos de la UNLP. Para cada alumno se ha registrado: nombre de la carrera (20 caracteres), legajo (10 caracteres), nombre y apellido (30 caracteres) y promedio (float).

a) **(2 puntos)** Escriba una función que genere un archivo de texto con la información correspondiente a los alumnos de una carrera dada. La función recibirá 3 parámetros: un puntero a FILE correspondiente al archivo binario, el nombre del archivo de texto a generar y el nombre de la carrera a la que pertenecerán los alumnos seleccionados. En el archivo de texto, los datos de cada estudiante (legajo, nombre y apellido y promedio) deben aparecer en una línea separados por coma.

```
struct ALUMNO {
   char carrera[20];
   char legajo[10];
   char nom[30];
   float promedio;
};
void alumnsCarrera(FILE * archB, char * nomTexto, char * nomCarr ){
    FILE * archT;
    struct ALUMNO alu;

    archT = fopen(nomTexto, "w");

    //el arch.está abierto por las dudas, nos paramos al inicio
    fseek(archB, 0, SEEK_SET);

    fread(&alu,sizeof(struct ALUMNO), 1, archB);

    while (!feof(archB)){
        if (strcmp(nomCarr, alu.carrera)==0)
        fprintf(archT, " %s - %s - %s  -  %.2f\n",
                alu.carrera, alu.legajo, alu.nom, alu.promedio);
        fread(&alu,sizeof(struct ALUMNO), 1, archB);
    }
    fclose(archT);
}
```

b) **(1 punto)** Implemente un programa que abra el archivo binario y genere el archivo de texto usando la función definida en a). Los nombres del archivo binario y del archivo de texto así como el de la carrera se ingresan por teclado. Si al finalizar el archivo de texto queda vacío (es decir que en el archivo binario no había ningún alumno de la carrera ingresada) debe mostrar el texto “SIN ALUMNOS”.

```
int main(){
    char nTexto[50], nBinario[50], nCarr[20];
    FILE * archBinario;

    printf("Nombre del archivo binario: ");
    scanf("%s", nBinario);

    printf("Nombre del archivo de texto: ");
    scanf("%s", nTexto);

    printf("Nombre de la carrera: ");
    scanf("%s", nCarr);

    archBinario = fopen(nBinario, "rb+");
    if (archBinario==NULL)
        printf("Error al abrir el archivo!");
    else{
        alumnsCarrera(archBinario, nTexto, nCarr);
        fclose(archBinario);
    }

    // Veamos si el archivo de texto está vacío
    FILE * archT;
    char linea[255];

    archT = fopen(nomTexto, "r");
    fgets(linea, 255, archT);
    if (feof(archT))
        printf("CARRERA SIN ALUMNOS!");
    fclose(archT);

    // esto ultimo también se puede hacer con ftell()
    // ubicándose al final del archive previamente
    return 0;
}
```

6. **(2.5 puntos)** Memoria dinámica.

Se desea procesar la información climática de cada día del año 2022. Para cada día se tienen 3 valores de tipo float: temperatura mínima, temperatura máxima y humedad.

Realice un programa que cargue la información en una matriz dinámica leyendo los valores de temperatura mínima, temperatura máxima y humedad desde teclado. Una vez inicializada la estructura, realice un módulo que imprima para cada mes: el nombre del mes y el número de día con mayor temperatura. Por último, libere la memoria reservada.

Nota: La reserva de memoria, la carga de información, el procesamiento y la liberación de memoria deben modularizarse.

```c
#include <stdio.h>
#include <stdlib.h>

struct CLIMA {
   float tMin;
   float tMax;
   float humedad;
};

char * MES[]={"Enero", "Febrero", "Marzo","Abril", "Mayo", "Junio",
"Julio", "Agosto","Septiembre","Octubre", "Noviembre", "Diciembre"};
int ndias[]={31,28,31,30,31,30,31,31,30,31,30,31};

struct CLIMA ** reservarMemoria(int, int *);
struct CLIMA ** reservarMemAlternativa(int, int);
void cargarDatos(struct CLIMA **, int, int *);
void liberar(struct CLIMA **, int);
void verDatos(struct CLIMA**, int, int *);
void diasHumedos(struct CLIMA **, int, int *);

int main(){
    struct CLIMA **M;
    M = reservarMemoria(12,ndias);
    cargarDatos(M, 12, ndias);
    verDatos(M, 12, ndias);
    diasHumedos(M, 12, ndias);
    liberar(M, 12);
    return(0);
}

void diasHumedos(struct CLIMA **m, int FIL, int * COLS){
   int mes, dia, diaMasHum=0;
    for (mes=0; mes < FIL; mes++){
        for (dia=0; dia<COLS[mes]; dia++)
           if (m[mes][dia].humedad> m[mes][diaMasHum].humedad)
               diaMasHum = dia;

        printf("%10s - dia %2d - Humedad %.2f\n",
               MES[mes],diaMasHum+1, m[mes][diaMasHum].humedad);
    }
}

struct CLIMA ** reservarMemoria(int FIL, int * COLS){
    struct CLIMA **mat;
    int mes;
    mat = malloc(FIL*sizeof(struct CLIMA *));
    for (mes=0; mes < FIL; mes++)
        mat[mes]= malloc(COLS[mes]*sizeof(struct CLIMA));

    return(mat);
}

struct CLIMA ** reservarMemAlternativa(int FIL, int COL){
    // el enunciado no dice que la reserva deba ser exacta
    struct CLIMA **mat;
    int mes;
```

```
    mat = malloc(FIL*sizeof(struct CLIMA *));
    // todos los meses tienen la misma cantidad de días
    for (mes=0; mes < FIL; mes++)
        mat[mes]= malloc(COL*sizeof(struct CLIMA));

    return(mat);
}

void cargarDatos(struct CLIMA **m, int FIL, int * COLS){
    struct CLIMA E;
    int mes, dia;
    for (mes=0; mes < FIL; mes++){
        printf("%s - Ingrese Temp.Min,Temp.Max y humedad:\n",MES[mes]);
        for (dia=0; dia<COLS[mes]; dia++){
           printf("dia %2d: ", dia+1);
           scanf("%f %f %f", &E.tMin, &E.tMax, &E.humedad);
           m[mes][dia] = E;
        }
    }
}

void liberar(struct CLIMA **mat, int FIL){
    int mes;
    for (mes=0; mes < FIL; mes++)
        free(mat[mes]);
    free(mat);
}

void verDatos(struct CLIMA** m, int FIL, int *COLS){
   struct CLIMA E;
   int mes, dia;
    for (mes=0; mes < FIL; mes++){
        printf("%s\n", MES[mes]);
        for (dia=0; dia<COLS[mes]; dia++){
           E = m[mes][dia];
           printf("dia %2d: %.2f %.2f %.2f\n", dia+1,
                   E.tMin, E.tMax, E.humedad);
        }
    }
}
```