Por favor siga las siguientes indicaciones:

- Escriba con lápiz.
- Escriba su nombre y número de documento en todas las hojas que entregue.
- Numere las hojas e indique el total de hojas en la primera de ellas.
- Escriba las hojas de un solo lado.
- Comience cada ejercicio en una hoja nueva.
- El total máximo de puntos del examen es 100.
- El examen contiene un total de: 2 páginas.
- La prueba es individual y sin material.
- Solo se contestan dudas acerca de la letra de los ejercicios.
- Duración 2 horas y media.

## Ejercicio 1 (15 puntos)

1. ¿Qué es el alcance de una variable? ¿Cuál es el alcance de una variable definida fuera de todas las funciones de un archivo (incluyendo el main)?

***Solución:***

El alcance de una variable es la parte de programa donde la variable es conocida y por lo tanto puede ser utilizada.
Una variable definida fuera de todas las funciones de un archivo (incluyendo el main) se llama variable global y su alcance son todas las funciones que se encuentren debajo de donde está declarada.

2. ¿Cómo funciona la estructura de selección múltiple switch?

***Solución:***

La estructura de selección múltiple switch funciona a partir de evaluar una expresión y para cada valor que pueda tomar ejecutar una secuencia de sentencias.
Por ejemplo:

```
int i;
printf("Ingrese un entero: ");
scanf("%d", &i);

switch(i) {
      case 0:
            //grupo sentencias cuando i=0
            break;
      case 1:
            //grupo sentencias cuando i=1
            break;
      case 2:
            //grupo sentencias cuando i=2
            break;

      default:
            //grupo sentencias cuando i tiene cualquier valor que no es 0, 1 o 2
            break;
}
```

En este ejemplo la expresión a evaluar es la variable *i*. De acuerdo al valor de *i*, es el grupo de sentencias que se van a ejecutar. El caso por defecto (*default*) se corresponde con todos los valores que no tengan un *case* más arriba. Luego de cada grupo de sentencias es necesario poner un *break*, dado que sino, se seguirían ejecutando las sentencias de los *case* que estén por debajo del correspondiente hasta encontrar un *break*, o hasta que se terminan los *case*.

**Ejercicio 2 (25 puntos)**

Una palabra es una secuencia ordenada de caracteres que puede ser vacía y que nunca puede tener más de 50 caracteres. Por tanto la estructura que modela la palabra podría ser la siguiente:

```
#define LARGO 50

struct palabra{
      char elems[LARGO];
      int pos;
};
```

Se utilizará el campo pos para indicar el último carácter en la palabra. Si la palabra es vacía pos= -1.

**Ejemplo:** Si palabra es “Hola” entonces elems = ['H','o','l','a'] y pos=3

Se pide implementar las siguientes funciones.

1. `palabra merge (palabra p1,palabra p2);`
   La función intercala las letras de p1 y p2, comenzando con las de p1. Si una palabra es más larga que la otra, las letras restantes de la palabra más larga se colocarán al final de la palabra.

   Ejemplo ilustrativo: merge(“aeiou”,”vwxyz”) = avewixoyuz
   merge(“mano”,”lechuga”) = mlaencohuga
   merge(“”,”oso”)= oso

***Solución:***

```
palabra merge (palabra p1,palabra p2){
        palabra res;
        res.pos = -1;//inicializo el resultado como una palabra vacia

        int i_p1 = 0;
        int i_p2 = 0;
        int i_res = 0;

        while ((i_p1 <= p1.pos) || (i_p2 <= p2.pos)){

           if (i_p1 <= p1.pos){
              res.elems[i_res] = p1.elems[i_p1];
              ++i_res;
              ++i_p1;
           }

           if (i_p2 <= p2.pos){
              res.elems[i_res] = p2.elems[i_p2];
              ++i_res;
              ++i_p2;
           }
        }

        res.pos = --i_res;//decremento porque en la ultima pasada i_res queda
                          //una posición adelante

        return res;
}
```

2. **`int length (palabra p);`**
   Calcula la longitud de una cadena de texto. Si la cadena es vacía devuelve 0.
   Se pide realizar dos implementaciones diferentes de la misma función.

***Solución:***

```
int length1(palabra p){
    return p.pos + 1;
}

int length2(palabra p){
    int i = 0;
    while (p.elems[i]!='\0'){
          ++i;
    }
    return i;
}
```

**Ejercicio 3 (20 puntos)**

1. Se desea implementar recursivamente la función *búsquedaLineal*, la cual realiza la búsqueda lineal (búsqueda posición a posición) de un entero *x* en un arreglo, devolviendo la posición del arreglo donde se encuentra *x*, o -1 en caso contrario.

   **`int busquedaLineal(int x, int[] arreglo, int largoArreglo)`**

   Nota: si necesita parámetros adicionales para generar la recursión, puede definir una función auxiliar que realice la recursión y tenga los parámetros que necesite y la invoca desde **`busquedaLineal`**.

***Solución:***

```
int busquedaLineal(int x, int arreglo[], int largoArreglo){

    if (largoArreglo == 0){
       return -1;//no hay elementos por lo tanto no está x
    }
    else {
         if (arreglo[largoArreglo -1] == x)
            return largoArreglo -1;
         else return busquedaLineal(x,arreglo,largoArreglo-1);
         }
}
```

2. Escriba un ejemplo de invocación de la función **`busquedaLineal`** desde el **`main().`**

   Deberá declarar y asignar valores a las variables que necesite para la invocación. Deberá indicar cuál es el valor que retornará **`busquedaLineal`** para el ejemplo planteado.

***Solución:***

```
int main(){

      int arreglo [2];
      arreglo[0]=10;
      arreglo[1]=20;
      arreglo[2]=30;

      int pos = busquedaLineal(20,arreglo,3);

      printf ("posicion=%d\n",pos);

      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Para este ejemplo **`busquedaLineal`** retorna 1.

**Ejercicio 4 (40 puntos)**

El centro de apuestas de caballos “HIDALGO” desea tener un sistema informático que le permita registrar las apuestas que realizan los apostadores. Para cada apuesta necesita registrar la siguiente información:

- Fecha de la carrera
- Cedula del apostador
- Nombre del caballo al que apuesta
- Monto de dinero que apuesta

Escribe un programa en C/C++ que despliegue y maneje el siguiente menú:

1- Alta de apuesta
2- Listar apuestas
0- Salir

En la **opción 1** se le pedirá al usuario que ingrese la fecha de la carrera, la cedula del apostador, el nombre del caballo al que apuesta y el monto de dinero que apuesta. El sistema dará de alta la apuesta y mostrará en pantalla el mensaje “Apuesta creada con éxito”.

En la **opción 2** se debe mostrar en pantalla la información de todas de las apuestas almacenadas.

En la **opción 0** el programa debe terminar.

Notas:

- La fecha de la apuesta se representará con una estructura de 3 campos de tipo entero (un campo para el día, otro para el mes, y otro para el año).
- La cédula se representará con un arreglo de caracteres de largo 8.
- El nombre del caballo se representará con un arreglo de caracteres de largo 20.
- El monto de dinero se representará con un entero.
- Se puede asumir que no se van a registrar más de 10.000 apuestas.
- Se pueden definir funciones y procedimientos auxiliares.

**Solución:**

```
#include <cstdlib>
#include <iostream>
using namespace std;

#define MAX_LARGO_CEDULA 8
#define MAX_LARGO_NOMBRE 20
#define MAX_CANT_APUESTAS 10000

//estructura para guardar una fecha
struct Fecha{
       int dia;
       int mes;
       int ano;
};

//estructura para guardar los datos de una apuesta
struct S_apuesta{
       Fecha fechaCarrera;
```

```
       char cedApostador[MAX_LARGO_CEDULA];
       char nomCaballo[MAX_LARGO_NOMBRE];
       int monto;
};


//array con tope para guardar todas las apuestas realizadas
struct Apuestas{
       S_apuesta apuestas[MAX_CANT_APUESTAS];
       int tope; //cantidad de apuestas realizados
};


int altaResultadoApuesta(Apuestas & apuestasRealizadas, S_apuesta apuesta){


       int posLibre = apuestasRealizadas.tope;
       apuestasRealizadas.apuestas[posLibre].fechaCarrera.dia =
apuesta.fechaCarrera.dia;
       apuestasRealizadas.apuestas[posLibre].fechaCarrera.mes =
apuesta.fechaCarrera.mes;
       apuestasRealizadas.apuestas[posLibre].fechaCarrera.ano =
apuesta.fechaCarrera.ano;

       strcpy(apuestasRealizadas.apuestas[posLibre].cedApostador,
apuesta.cedApostador);
       strcpy(apuestasRealizadas.apuestas[posLibre].nomCaballo, apuesta.nomCaballo);
       apuestasRealizadas.apuestas[posLibre].monto = apuesta.monto;

       ++ apuestasRealizadas.tope;//aumento la cantidad de apuestas

    return posLibre;
}

void listarApuestas(Apuestas apuestasRealizadas){

     if (apuestasRealizadas.tope == -1)
        printf("No existen apuestas.\n");
     else {
          int pos = 0;
          while(pos <= apuestasRealizadas.tope -1){
            printf("---Datos de la apuesta %d\n", pos+1);
            printf("Fecha de la carrera:
%d/%d/%d\n",apuestasRealizadas.apuestas[pos].fechaCarrera.dia,
apuestasRealizadas.apuestas[pos].fechaCarrera.mes,
apuestasRealizadas.apuestas[pos].fechaCarrera.ano);
            printf("Cedula del apostador: %s\n",
apuestasRealizadas.apuestas[pos].cedApostador);
            printf("Nombre del caballo:
%s\n",apuestasRealizadas.apuestas[pos].nomCaballo);
            printf("Monto de la apuesta:
%d\n",apuestasRealizadas.apuestas[pos].monto);
            ++pos;
          }
     }
}


int main()
{

    Apuestas apuestasRealizadas;
    apuestasRealizadas.tope = 0;
```

```
    int opcion;
    Fecha fechaApuesta;
    S_apuesta apuesta;
    int res;

    do {
        printf("Menu de opciones\n");
        printf("1- Alta de apuesta\n");
        printf("2- Listar apuestas\n");
        printf("0- Salir\n");
        printf("Ingrese la opcion deseada: ");
        scanf("%d",&opcion);

        switch (opcion){
               case 1:
                    printf("Ingrese la fecha de la carrera en formato dia/mes/ano:
");

scanf("%d/%d/%d",&apuesta.fechaCarrera.dia,&apuesta.fechaCarrera.mes,&apuesta.fechaCa
rrera.ano);
                    printf("Ingrese la cedula del apostador (sin digito verificador):
");
                    scanf("%s",apuesta.cedApostador);
                    printf("Ingrese el nombre del caballo: ");
                    scanf("%s",apuesta.nomCaballo);
                    printf("Ingrese el monto de la apuesta: ");
                    scanf("%d",&apuesta.monto);

                    res = altaResultadoApuesta(apuestasRealizadas,apuesta);

                    printf("Apuesta numero %d agregado con exito.\n",res+1);

                    break;
               case 2:
                    listarApuestas(apuestasRealizadas);
                    break;
        }
    }while (opcion!=0);

    system("PAUSE");
    return EXIT_SUCCESS;
}
```