PREGUNTA 1 (2 puntos).

Disponemos de una matriz de N x M, siendo tanto N como M mayores que 5, rellena de caracteres. Debes comprobar mediante la construcción de un módulo si aparecen en la misma fila 4 caracteres seguidos iguales. En ese caso deberás devolver (no escribir) el número de la fila en la que aparecen.

Ejemplo: si se diese como entrada la siguiente:

*	-	#	#	*	.	3
6	t	¿	*	*	*	-
#	#	#	#	-	2	r
*	f	e	*	*	*	-
e	$	,	&	r	#	#

Devolvería el número de fila **2**

Importante: No puedes utilizar estructuras auxiliares.

PREGUNTA 2. (2 puntos)

Realiza una función recursiva en la que a partir de un número entero devuelva el número de veces que aparecen las cifras 3 y 5. La función contará con un único parámetro de entrada.

Por ejemplo:

contando(3532) devuelve 3
contando(705) devuelve 1
contando(1682) devuelve 0

Importante: este ejercicio sólo puntuará si se realiza de forma recursiva.
Importante: No se podrán utilizar arrays ni registros para almacenar los datos.

PREGUNTA 3. (2 puntos)

Realiza un módulo que permita encriptar un número entero, es decir, transformarlo en otro a partir del siguiente algoritmo: cada dígito se sustituye por la suma las cifras de su codificación en ASCII, pero si este nuevo valor tiene varias cifras, se sumarán de nuevo estas cifras hasta conseguir un solo dígito. El carácter '0' en ASCII corresponde al número entero 48.

Por ejemplo:

encriptar(0) dará como resultado: 3 (El 0 es el 48, 4+8 = 12 y 1+2=3)

y del mismo modo, es decir, dígito a dígito sería:

encriptar(4675) dará como resultado: 7918

encriptar(11) dará como resultado: 44

Importante: No se podrán utilizar arrays ni registros para almacenar los datos.

PREGUNTA 4. (1+1+2 puntos)

Disponemos de la información de 300 deportistas. De cada uno de ellos nos interesa almacenar su dorsal, nombre, DNI, altura, tipo de deporte que practica (A: atletismo, N: natación, T: tenis) y posición en la que ha quedado en cada una de las pruebas que ha realizado. Cada deportista sólo puede practicar un deporte a la vez. Un deportista puede haber participado **como máximo** en 15 pruebas.

a) Diseña las estructuras de datos adecuadas para almacenar dicha información.
b) Diseña un módulo que muestre en pantalla el nombre y DNI de un deportista a partir de su dorsal.
c) Diseña uno o varios módulos que muestren los nombres de los deportistas que han quedado en las tres primeras posiciones de cada uno de los deportes. Se valorará la reutilización del mismo módulo para mostrar los primeros clasificados de cada deporte.

Puedes hacer uso de las funciones `strcpy()` y `strcmp()` **si lo crees conveniente**, no es obligatorio.

- Para copiar dos cadenas puedes usar
  `strcpy(cadena_destino, cadena_origen).`
- Para comparar dos cadenas puedes usar `strcmp`. Si `strcmp(cadena1, cadena2)` devuelve un 0 significa que ambas cadenas son iguales.

Soluciones:

//ejercicio 1

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


void mostrar (char m[][7]){
    int f, c;

    for (f=0; f<5; f++) {
         for (c=0; c<7; c++)
              cout << m[f][c] << " ";
         cout << endl;
    }
}

int repetidos(char m[][7]){
  int f, c, repe, fila;
  char car;

  fila = -1; // por si no aparece ningún carácter cuatro veces seguidas
  for (f=0; f<5; f++) {
    repe = 0;
    car = m[f][0];
    for (c=1; c<7; c++){
      if (car == m[f][c])
         repe++;
      else {
         repe = 0;
         car = m[f][c];
      }
       if (repe>=3)
         fila = f;
    }
  }
  return(fila);
}

int main(){
    int fila;
  char matriz[5][7]= { '*', '-', '#', '#', '*', '.', '3',
                       '6', 't', '?', '*', '*', '*', '-',
                       '#', '#', '#', '#', '-', '2', 'r',
                       '*', 'f', 'e', '*', '*', '*', '-',
                       'e', '$', ',', '&', 'r', '#', '#'};

  mostrar(matriz);
  fila = repetidos(matriz);
  if (fila != -1)
    cout << "Aparecen más de 4 caracteres iguales seguidos en la fila " <<
fila << endl;
  else
    cout << "No aparecen más de 4 caracteres iguales seguidos en la matriz."
<< endl;
}
```

//ejercicio 2

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

int contando(int n){

  int res;

  res = 0;

  if (n>0){
    if (n%10 == 3 || n%10 == 5)
         res = 1 + contando(n/10);
    else
         res = contando(n/10);
  }
  return(res);
}

int main(){
   int num, veces;

    cout << "Introduce un número: ";
    cin >> num;

  veces = contando(num);
  if (veces != 0)
    cout << "Las cifras 3 y/o 5 aparecen " << veces << " en el número " <<
num << endl;
  else
    cout << "Las cifras 3 y/o no aparecen en el número " << num << endl;
}
```

//ejercicio 3

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

int encriptar(int num){
  int valor, cifra, nuevonum, suma, mult;

  if (num != 0)
    nuevonum = 0;
  else
    nuevonum = 3;   // para el caso en el que el número sea el 0

  suma = 0;
  mult = 1;
  while (num > 0) {
    cifra = num % 10;
    valor = cifra + 48;
    suma = 0;
    while (valor>0) {
      suma = suma + valor%10;
      valor = valor/10;
      if (valor == 0 && suma > 9) {
         valor = suma;
         suma = 0;
      }
    }
    nuevonum = suma * mult + nuevonum;
    mult = mult*10;
    num = num/10;
  }
  return(nuevonum);
}

int main() {
  int num, numencrip;

  cout << "Número: ";
  cin >> num;

  numencrip = encriptar(num);

  cout << "El número " << num << " encriptado es " << numencrip << endl;
}
```

//ejercicio 4

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

const int kmax = 300;

typedef char Tcadena[20];
typedef char Tdni[10];
typedef struct {
                        int dorsal;
                        Tcadena nombre;
                        Tdni dni;
                        float altura;
                        char deporte;  // A: atletismo, N: natación, T: tenis)
                        int numpruebas;
                        int pruebas[15];
                        float media;
} Tdeportista;

typedef Tdeportista Tdeportistas [kmax];


void mostrarDeportista(Tdeportista depor) {
     int j;

     cout << "---------" << endl;
     cout << "Dorsal: " << depor.dorsal << endl;
     cout << "Nombre: " << depor.nombre[0] << endl;
     cout << "DNI: " << depor.dni[0] << endl;
     cout << "Altura: " << depor.altura << endl;
     cout << "Deporte: " << depor.deporte << endl;
     cout << "Núm. de pruebas " << depor.numpruebas << " : " ;
     for (j=0; j<depor.numpruebas; j++)
          cout << depor.pruebas[j]  << " , " ;
     cout << endl;
     cout << "Media de sus pruebas: " << depor.media << endl;
     cout << "---------" << endl;
     cout << endl;
}

void mostrarPrimeros(Tdeportista depor, int total){
  int i, media, suma;
  int mejores[3] = {0, 0, 0};

  suma = 0;     // he decidido calcular la media de sus posiciones para
establecer el orden

  // Hay formas más eficientes, como ordenar los deportistas por su media
  // para facilitar el proceso haré tres pasadas para encontrar cada posición
  mejores[0] = depor[0].media;
  for (i=0; i<total; i++) {
    if (depor[i].media < mejores[0]) {
           mejores[0] = depor[i].media;
           pos = i


}

void rellenar(Tdeportistas depor, int total) {
```

```
      int i, j, deporte, suma;
      float media;

      for (i=0; i<total; i++) {
         depor[i].dorsal = rand()%300+1;
         depor[i].nombre[0] = 'a' + rand()%24;
         depor[i].dni[0] = '0' + rand()%8;
         depor[i].altura = 1 + (rand()%10)*0.8;
         deporte = rand()%3;
         switch(deporte) {
              case 0: depor[i].deporte = 'A'; break;
              case 1: depor[i].deporte = 'N'; break;
              case 2: depor[i].deporte = 'T';
         }
         depor[i].numpruebas = rand()%15 +1;
         suma = 0;
         for (j=0; j<depor[i].numpruebas; j++) {
              depor[i].pruebas[j]   = rand()%300 +1;
              suma = suma + depor[i].pruebas[j];
         }
         media = suma / depor[i].numpruebas;
         depor[i].media = media;

      }

      mostrarDeportista(depor[4]);

   }

   int buscarDeportista (Tdeportistas depor, int dorsal, int total) {
       int i, pos;

       pos = -1;
       i=0;
       while (i < total && pos == -1) {
         if (depor[i].dorsal == dorsal)
           pos = i;
         i++;
       }
       return(pos);   // devolverá la posición en el registro del deportista con
ese dorsal
                                    // si no lo encuentra, devolverá -1
   }

   int main() {
     Tdeportistas deportistas;
     int pos, dorsal, total;

     srand(time(NULL));
     total = 20; // voy a introducir aleatoriamente 20 deportistas
     rellenar(deportistas, total);

     cout << "Número de dorsal: ";
     cin >> dorsal;
     pos = buscarDeportista(deportistas, dorsal, total);
     if (pos == -1)
       cout << "No hay ningún deportista con el dorsal " << dorsal << endl;
     else {
       cout << "El deportista con el dorsal " << dorsal << " es " <<
deportistas[pos].nombre << endl;
       cout << "con DNI " << deportistas[pos].dni << endl;
     }
}
```