GUIA DE EJERCICIOS

Wenceslao Palma <wenceslao.palma@ucv.cl>

1 Iteración, selección

1. Escriba un programa en lenguaje C que genere las primeras N filas de:

```
      1
    2 3 2
  3 4 5 4 3
4 5 6 7 6 5 4
    ...
```

2. Un número entero se puede expresar como un producto de números primos. Dicha expresión se llama descomposición de un número en factores primos. Escriba un programa en lenguaje C que calcule la descomposición de un número en factores primos. Por ejemplo: $13860 = 2 \times 2 \times 3 \times 3 \times 5 \times 7 \times 11$

3. El algoritmo de Newton, usado para calcular la raíz cuadrada, se basa en aproximaciones sucesivas donde la primera aproximación de la raíz de un número N es:
   $T_1 = N/2$
   $T_{i+1} = T_i/2 + N/(2 * T_i)$

   Escriba un programa en lenguaje C, que permita determinar la raíz cuadrada de un número en base a la recurrencia anterior.

4. Dado un número entero positivo, su crápulo es un número que se obtiene de la sgte forma: se suman los dígitos que lo componen y si el valor de la suma es menor que 10, el crápulo es el valor obtenido sino el crápulo es el crápulo de la suma de los dígitos. Escriba un programa en lenguaje C que lea un entero positivo y escriba el valor de su crápulo.

5. Un número entero n se dice perfecto si la suma de sus divisores propios es igual a n. Si la suma de los divisores es mayor que n, el número se dice abundante. Si dicha suma es menor que n, el número se dice deficiente. Escriba un algoritmo que determine si un número es abundante, deficiente o perfecto.

6. Cuando la suma de los dígitos alternos de un número son iguales ó múltiplo de 11, dicho número es divisible por 11. Por ejemplo: 5841. 5+4=8+1. Escriba un programa en lenguaje C que determine si un número entero positivo es divisible por 11.

7. Un número se dice automórfico si su cuadrado termina en los mismos dígitos que el número original, por ejemplo $76^2 = 5776$. Un número se dice trimórfico si su cubo termina en los mismos dígitos que el número original, por ejemplo $49^3 = 117649$. Escriba un programa en lenguaje C que determine los números automórficos y trimórficos menores que 1000.

8. La multiplicación rusa consiste en multiplicar sucesivamente por 2 el multiplicando y dividir por 2 el multiplicador hasta que el multiplicador tome el valor 1. Luego, se suman todos los multiplicandos correspondientes a los multiplicadores impares. Dicha suma es el resultado del producto de los dos números. Por ejemplo, para multiplicar 37 por 12 el resultado final es $12 + 48 + 384 = 444$.

9. Escriba un programa en lenguaje C que permita obtener la suma de todos los números que se puedan generar al permutar n dígitos consecutivos a partir del 1. El número de permutaciones se define como todos los números distintos que se pueden formar cambiando la posición de los dígitos. La cantidad de permutaciones de números con n dígitos es $n! = 1 \times 2 \times 3 \times 4 \times ..... \times n$. El siguiente algoritmo permite realizar el proceso en forma rápida:

   - Cada dígito estará $n!/n$ veces en la misma posición en el total de números obtenidos de las permutaciones.
   - A continuación se suman los n dígitos y se multiplica este resultado por el número de veces obtenido en el paso anterior.
   - El número obtenido en el paso 2 se suma n veces, siendo n el número de dígitos. Cada sumando debe desplazarse una posición.

   Ejemplo: Considerando n=3, el número será 123. El número de veces que se repite cada dígito en la misma posición es $3!/3 = 2$. La suma de los dígitos es $1 + 2 + 3 = 6$. La multiplicación de la suma de los dígitos por el número de veces que se repite cada dígito es $2 \times 6 = 12$. Como son 3 dígitos debe sumarse 3 veces 12 con un desplazamiento cada vez, es decir:

```
    12
   12
+ 12
 -------
  1332
```

10. Un número entero positivo se dice M-alternante si: El dígito menos significativo es par (impar) entonces los 2 dígitos siguientes deben ser impares (pares), luego los tres siguientes dígitos deben ser pares (impares) y así sucesivamente. Por ejemplo : 357221 es un número M-alternante de orden 3. Escriba un programa que determine si un número es o no M-alternante y si lo es a que orden corresponde.

11. Un anillo primo esta compuesto de n dígitos (n par), ubicados uno al lado del otro, y es tal que la suma de los dígitos adyacentes siempre es un número primo. Por ejemplo : 143256 corresponde a un anillo primo ya que:

```
1+4 = 5
4+3 = 7
3+2 = 5
2+5 =7
5+6 = 11
6+1 = 7
```

Escriba un programa en C que verifique si un número ingresado cumple con la condición de estar compuesto por una serie de dígitos que lo califican como un anillo primo.

12. Un par de números m y n son llamados par amigable, si la suma de todos los divisores de m (excluyendo m) es igual al número n y la suma de todos los divisores del número n (excluyendo n) es igual a m ($m! = n$). Por ejemplo, los numeros 220 y 284 son un par amigable porque los únicos números que dividen de forma exacta a 220 son 1,2,4,5,10,11,20,22,44,55 y 110, y $1 + 2 + 4 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$. Los números que dividen a 284 son 1,2,4,71 y 142, y $1 + 2 + 4 + 71 + 142 = 220$. Por lo tanto 220 y 284 son un par amigable. Escriba un programa en C que ingrese m y n, asegure que m es distinto de n y determine si dichos números son par amigable.

13. Cuando un número es expresado en binario su k-ésimo dígito representa un múltiplo de 2^k. Por ejemplo:

$$10011_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 19$$

Por otro lado, cuando un número es expresado en base sb su k-ésimo dígito representa un múltiplo de $2^{k+1} - 1$. Los números se componen únicamente de 0s y 1s excepto el dígito menos significativo distinto de 0 el cual puede ser un 2. Por ejemplo:

$$10120_{sb} = 1\times(2^5-1)+0\times(2^4-1)+1\times(2^3-1)+2\times(2^2-1)+0\times(2^1-1) = 44$$

Escriba un programa en lenguaje C que:

- verifique que un número se encuentre expresado en base sb.
- en caso que el número se encuentre en base sb entregue su equivalente en decimal.

14. Un número es polidivisible si es divisible por su longitud y, además, si le quitamos el último dígito resulta un número que a su vez es polidivisible. Por ejmplo, el número 381.654.729 es polidivisible ya queda

```
381.654.729  = 9 x 42.406.081
 38.165.472  = 8 x  4.770.684
  3.816.547  = 7 x    545.221
    381.654  = 6 x     63.609
     38.165  = 5 x      7.633
      3.816  = 4 x        954
        381  = 3 x        127
         38  = 2 x         19
          3  = 1 x          3
```

Existen otros números polidivisibles como el 102 y el 9.876 pero su cantidad es limitada (hay un total de 20.456 números polidivisibles el mayor de los cuales tiene 25 dígitos). Escriba un programa en lenguaje C que determine si un número dado es o no es polidivisible.

15. En el presente ejercicio, Ud. debe codificar un programa en C que calcule la máxima cantidad de números (excluyendo el 1) que pueden ser utilizados para expresar $n!$. Por ejemplo:

    $8! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 = 2 \times 3 \times 2 \times 2 \times 5 \times 2 \times 3 \times 7 \times 2 \times 2 \times 2 = 2^7 \times 3^2 \times 5 \times 7$

    en este caso, la máxima máxima cantidad de números (excluyendo el 1) que pueden ser utilizados para expresar 8! es 11.

    Algunos ejemplos:

Entrada	Salida
5	5
9	13
1996	5957
123456	426566

2 Arreglos unidimensionales/bidimensionales

1. Desarrolle un programa en lenguaje C que permita rotar una matriz en ángulos de 0,90,180 ó 270 grados. El programa debe: ingresar valores en una matriz de NxN, el ángulo en que se desea rotar y mostrar la matriz rotada en el ángulo que corresponda. Por ejemplo:

```
1 2 rotación en 90° 2 4
3 4                 1 3
```

2. En un tablero de N filas y M columnas ($1 \leq N, M \leq 100$) se encuentran todos los números entre 1 y $N \times M$. Con el objetivo de ordenar el contenido del tablero, en orden ascendente por fila, el único movimiento permitido es invertir el contenido de una fila o de una columna. Escriba un programa en C que dado un tablero verifique si un tablero se encuentra ordenado luego de realizar una cierta cantidad de movimientos de fila y/o columna. En la figura se tiene un tablero de 4×4 el cual queda totalmente ordenado luego de realizar movimientos sobre la tercera fila (F3), la tercera columna (C3) y la segunda fila (F2).

```
1  2  15  4     F3     1  2  15  4     C3     1  2  3  4     F2     1  2  3  4
8  7  11  5     --->   8  7  11  5     --->   8  7  6  5     --->   5  6  7  8
12 6  10  9            9  10 6   12           9  10 11 12           9  10 11 12
13 14 3   16           13 14 3   16           13 14 15 16          13 14 15 16
```

Datos de entrada: la primera línea de entrada contiene N y M separados por un espacio. Luego, la siguientes N líneas contienen M números cada una. Por último, se especifica una serie de movimientos especificados por un caracter $\in \{F, C\}$ seguido por el número de fila y/o columna sobre la cual se desea realizar un movimiento. Los movimientos terminan cuando se encuentra el par Z 0.
Salida: el programa debe mostrar el mensaje "Tablero ordenado" ó "Tablero desordenado" según corresponda.
Ejemplo:

```
Entrada          |     Salida
4 4              |      Tablero ordenado
1  2  15  4      |
8  7  11  5      |
12 6  10  9      |
13 14 3   16     |
F 3              |
C 3              |
F 2              |
Z 0              |
```

3. Un número se dice circular si es posible recorrerlo de la siguiente manera: Se toma el dígito de más a la izquierda y su valor indicará la cantidad de veces que se debe contar hacia la derecha, esta "cuenta" se detiene en un dígito del número el cual también indicará la cantidad de veces que se debe contar hacia la derecha y así sucesivamente. Este proceso da origen a un número circular si todos los dígitos del número se consideran tan solo

una vez (como número originador de un conteo) y además se retorna al dígito de más a la izquierda que fue el que comenzó el conteo.
Valide que no existan dígitos repetidos.
Ejemplo : Si el número es 81362

(1) **8** 1 3 6 2
(2) **8** 1 3 **6** 2
(3) **8** 1 3 **6 2**
(4) **8 1** 3 **6 2**
(5) **8 1 3 6 2**
(6) **8 1 3 6 2**

Este es un número circular!. Escriba un programa en lenguaje C que permita decidir si un número es circular.

4. Considere una secuencia arbitraria de números enteros. Es posible ubicar los operadores + y - entre los números de la secuencia, de este modo se obtienen diferentes expresiones aritméticas que producen diferentes valores. Por ejemplo para la secuencia: 17, 5, -21, 15 existen 8 posibles expresiones:

```
17 + 5 + -21 + 15 = 16
17 + 5 + -21 - 15 = -14
17 + 5 - -21 + 15 = 58
17 + 5 - -21 - 15 = 28
17 - 5 + -21 + 15 = 6
17 - 5 + -21 - 15 = -24
17 - 5 - -21 + 15 = 48
17 - 5 - -21 - 15 = 18
```

una expresión se dice divisible por K si lo operadores + y - son ubicados de tal manera que dan origen a un valor divisible por K. Escriba un programa que determine la divisibilidad de una expresión. Los datos de entrada deben ser K ($2 \leq K \leq 100$) y la secuencia de N enteros ($1 \leq N \leq 1000$). La salida debe ser: La expresión xxxxxx es Divisible por K o ninguna expresión generada es divisible por K.

Por ejemplo:

```
Entrada:  K=3 y la secuencia 17 5 21 15

Salida:
La expresión 17 - 5 + -21 + 15 = 6 es divisible por 3
La expresión 17 - 5 + -21 - 15 = -24 es divisible por 3
La expresión 17 - 5 - -21 + 15 = 48 es divisible por 3
La expresion 17 - 5 - -21 - 15 = 18 es divisible por 3
```

5. Un método clásico para encontrar todos los números primos, que se encuentran en una secuencia de números desde 2 hasta N, es la llamada Criba de Eratóstones. Para ello el método va marcando (eliminando) todos los múltiplos de 2, 3, 4, 5 y asi sucesivamente hasta que se encuentre el primer número no marcado que es mayor que $\sqrt{N}$.
Ejemplo : suponga una secuencia desde 2 a 20.

- Se toma el número 2, ya que es el primer número no marcado.
- Marcar todos los múltiplos de 2 a partir de 2^2.
- Se toma el 3 y se marcan todos los múltiplos de 3 a partir de 3^2.
- Se toma el 5 pero como es mayor que 20 se detiene el proceso y todos los números de la secuencia que no fueron marcados corresponden a los números primos que existen entre 2 *y* 20.

Escriba un programa en C que mediante el uso de arreglos y el método descrito anteriormente determine todos los números primos entre 2 y N.

6. Considere un juego que contempla un jugador y un tablero de MxN casilleros. En cada casillero es ubicado un entero positivo en el rango de 0 hasta 9. El objetivo del juego es remover todos los números desde el tablero. El jugador realiza sus intentos seleccionando un casillero, de este modo todos los casilleros ubicados en la **región conectada** que contienen el mismo número que el casillero seleccionado son eliminados y todos los casilleros ubicados sobre los eliminados son reubicados hacia abajo del tablero. Si todos los casilleros de una columna han sido removidos entonces las columnas ubicadas hacia su derecha se desplazan hacia la izquierda. El juego termina cuando todas los casilleros son eliminados o cuando no es posible continuar el juego.
Una **región conectada** esta compuesta de todas los casilleros adyacentes que pueden ser alcanzados mediante un movimiento horizontal (izquierda o derecha) y/o vertical (arriba o abajo) y considerando que todos los casilleros en la **región conectada** contienen el mismo número. Por ejemplo:

```
       1 3 5 2 2
       2 2 3 5 1
       1 2 3 5 5
Puede dar origen a las siguientes jugadas:

       1 3 5
       2 2 3 5 1
       1 2 3 5 5

           5
       1   3 5 1
       1 3 3 5 5

       1   5 1
       1 5 5 5

       1
       1 1
```

Se debe indicar como dato de entrada la dimension del tablero, indicando el número de filas y columnas. El contenido del tablero consiste de enteros no negativos $\in$ [0..9] que serán almacenados por filas, desde la posición

(0,0). Considere que un tablero puede tener un tamaño máximo de 10x10. Además el usuario, debe ingresar cada jugada en términos de coordenadas relativas al tablero. Las coordenadas deben ser ingresadas usando el siguiente formato: fila columna
Cada vez que el usuario juegue (ingresando una coordenada) se deberá mostrar el tablero resultante.

7. En un arreglo bidimensional llamado Puzzle se encuentra una instancia de un puzzle 4×4. Para determinar que tan lejos nos encontramos de la solución se puede utilizar la sgte expresión:

$$D = \sum_{i=0}^{15} |filaOriginal_i - filaActual_i| + |columnaOriginal_i - columnaActual_i|$$

   Escriba un programa en C que permita calcular D. No se permite el uso de arreglos auxiliares.

8. En un string (de largo máx 80 caracteres) se define el número de inversiones como el número de pares de caracteres que se encuentran desordenados uno respecto del otro. Por ejemplo: considerando un orden ascendente, en la secuencia de caracteres "$DAABEC$", el valor de la medida es 5 ya que el caracter D es mayor que 4 caracteres ubicados a su derecha y el caracter "E" es mayor que un caracter ubicado a su derecha. Escriba un programa en lenguaje C que calcule el número de inversiones para n strings. La entrada de strins termina cuando se ingresa el string "XXX".

9. La transformación de Burrows-Wheeler (BWT) es un algoritmo usado en técnicas de compresión de datos como bzip2. Para el texto de entrada **BANANA**, BWT funciona de la siguiente manera:
   Paso 1. Sea t ($1 \leq t \leq 100$) el tamaño del texto de entrada (en el ejemplo $t = 6$). Se genera una matriz M de dimensiones $t \times t$. La primera línea de M corresponde al texto de entrada. La i-ésima línea de M ($2 \leq i \leq t$) corresponde a un desplazamiento hacia la izquierda de $i-1$ posiciones con respecto al texto de entrada. En el caso de **BANANA** se obtiene la sgte matriz M (ver figura izq).

   Paso 2. Se ordenan las líneas de M en orden alfabético para obtener una nueva matriz P (ver figura derecha).

	1	2	3	4	5	6
1	B	A	N	A	N	A
2	A	N	A	N	A	B
3	N	A	N	A	B	A
4	A	N	A	B	A	N
5	N	A	B	A	N	A
6	A	B	A	N	A	N

	1	2	3	4	5	6
1	A	B	A	N	A	N
2	A	N	A	B	A	N
3	A	N	A	N	A	B
4	B	A	N	A	N	A
5	N	A	B	A	N	A
6	N	A	N	A	B	A

Paso 3. La BWT de **BANANA** corresponde al contenido de la última columna de P y al número de la fila de P donde se encuentre el texto de entrada. En el ejemplo, la última columna de P es **NNBAAA** y la fila que contiene **BANANA** es la 4. Por lo tanto, la BWT de **BANANA** es el par **(NNBAAA,4)**. Escriba un programa en lenguaje C que muestre la BWT dado un texto de entrada.

3 Funciones

1. Un número es un palíndrome si la secuencia de dígitos que lo componen es idéntica al leerla desde izquierda a derecha y viceversa. Ahora, al sumar un número como por ejemplo 65 con el número que da como resultado el leerlo desde la derecha hacia la izquierda se genera 121, el cual es palíndrome. Si se realiza lo mismo con otro número, llegará un momento en el cual se generará un palíndrome.

   Escriba la función **int cantidadSumas(long int)** que dado un número entero, entregue como resultado la cantidad de sumas necesarias para generar un palíndrome.

   Por ejemplo:

   ```
   Si n=47
   47+74 = 121 --> 1
   Si n =143
   143+341 = 484 --> 1
   Si n=87
   87+78 = 165
   165+561 = 726
   726+627 = 1353
   1353+3531 = 4884 --> 4
   ```

2. Construya la función int compara(char *,char *), la cual permite comparar 2 strings lexicográficamente. Los valores de retorno deben ser los mismos que los entregados por la función strcmp perteneciente al lenguaje C.

3. Desarrolle dos funciones en lenguaje C que permitan encontrar el valor de π mediante la siguiente expresión:

$$\frac{\pi}{2} = \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \frac{8}{7} \times ....$$

   (a) (10 ptos.) Una función considera que la precisión del cálculo depende de la cantidad de términos de la multiplicatoria: se consideran los n primeros términos, donde n es un parámetro de la función.

   (b) (15 ptos.) La otra función considera que la precisión del cálculo depende de la diferencia entre dos aproximaciones consecutivas del valor de π: si esta es menor que el parámetro de la función ϵ se detiene el cálculo.

   Ayuda: el i-ésimo término se puede generar usando i. Por ejemplo el **cuarto** término se puede generar usando $\frac{4}{4+1}$.

4. Considere que el arreglo bidimensional donde se almacena el texto es definido de la sgte manera: **char texto[10][100];** y las palabras se encuentran separadas por un espacio.

   Escriba la función

   int diferentes(char texto[][numcols],int numFils,int numCols); que retorna la cantidad de palabras diferentes dentro del texto.

5. Un número de nueve dígitos se dice deleitable si:

   (a) contiene exactamente los dígitos entre 1 y 9, una vez cada uno, y

   (b) los números creados tomando los n primeros dígitos ($1 \leq n \leq 9$) son cada uno de ellos divisibles por n, de tal forma que el primer dígito será divisible por 1 (siempre lo será), los dos primeros dígitos forman un número divisible por 2, los tres primeros forman un número divisible por 3 y así sucesivamente. Por ejemplo: si consideramos el número 123456789.

   ```
   1:1   = 1
   12:2  = 6
   123:3 = 41
   ```

   sin embargo 1234 no es divisible por 4. Un número deleitable es el 381654729.

   Escriba la función **int deleitable(int)** que permite determinar si un número es o no deleitable.

6. El Rey Arturo desea construir su mesa redonda en una sala donde toda la superficie de la mesa reciba la luz del sol. La sala escogida tiene en el techo una ventana triangular y el Rey Arturo desea construir la mesa más grande que pueda ser iluminada completamente por el triángulo de luz. Su tarea es escribir un programa en lenguaje C (usando structs y funciones) que ayude al Rey Arturo a encontrar el radio de la mesa más grande que es iluminada en forma completa por el triángulo de luz. Considere que la mesa es un círculo perfecto.

   Para el desarrollo de la tarea utilice la siguiente definición:

   ```
   typedef struct triangulo{
     double a;
     double b;
     double c;
   } triangulo;
   ```

Entrada

Los datos de entrada se componen de tres números a, b y c los cuales representan los lados del triángulo formado por la luz del sol. Ningún lado del triángulo será mayor que 1000000 y se puede asumir que $max(a, b, c) \leq (a+b+c)/2$. La presencia del valor -1 marca el fin de los datos de entrada.

Salida

Para cada caso de prueba se debe escribir en pantalla lo siguiente:

```
El radio de la mesa redonda es : r
```

donde r es el radio (redondeado a 3 dígitos) de la mesa redonda más grande que puede ser ubicada dentro del triángulo formado por la luz del sol.

Ejemplo

Dados los siguientes datos de entrada:

```
12.0 12.0 8.0
0.0 0.0 0.0
1.0 1.0 1.0
1.0 1.0 0.0
1.0 0.0 1.0
0.0 1.0 1.0
3.1416 3.1416 3.1416
1000000.0 1000000.0 1000000.0
1000000.0 500000.0 500000.0
0.1 0.1 0.1
3 4 5
30 40 50
300 400 500
3000 4000 5000
30000 40000 50000
300000 400000 500000
-1
```

La salida es:

```
El radio de la mesa redonda es : 2.828
El radio de la mesa redonda es : 0.000
El radio de la mesa redonda es : 0.289
El radio de la mesa redonda es : 0.000
El radio de la mesa redonda es : 0.000
El radio de la mesa redonda es : 0.000
El radio de la mesa redonda es : 0.907
El radio de la mesa redonda es : 288675.135
El radio de la mesa redonda es : 0.000
El radio de la mesa redonda es : 0.029
El radio de la mesa redonda es : 1.000
El radio de la mesa redonda es : 10.000
El radio de la mesa redonda es : 100.000
El radio de la mesa redonda es : 1000.000
El radio de la mesa redonda es : 10000.000
El radio de la mesa redonda es : 100000.000
```

4 Ruteos

Realice el ruteo de los sgtes programas mostrando el contenido de las variables dentro de cada función y la salida.

1.
```
int n=76;
int f(int *);

main(){
    int i=500;
    char c;

    c = f(&i);
    printf(''%d %d %d'',c,i,n);
}

int f(int *q){
    *q = ++n + *q;
    return *q;
}
```

2.
```
int r=9;
int f(int);

main(){
   int j=5,r;

   r = f(j);
   printf(''%d'',r,j);
}
int f(int x){
   x = x + r;
   return x;
}
```

3.
```
main(){
  int a, f=4,z;

  a = 'c'-1 == 'a';
  z = a * 2 + f%2 * 2;
  printf(''%d %d'',a,z);
```

4.
```
#define n 2
```

```
main(){
  int a[n][n]={1,2},{3,4}};
  int r;
  printf(‘‘%d’’, f(a));
}

int f(a[][n]){
  int i,j,s=0;

  for(i=0;i<=n-1;i++)
    for(j=0;j<=n-1;j++)
      s+=a[i][j];
 return s;
}
```

5 Estructuras

1. Dada una lista de rectángulos y una lista de puntos en el plano XY. Escriba un programa en lenguaje C que determine para cada punto cuales son los rectángulos que los contienen. Los puntos se encuentran descritos por sus coordenadas x e y y los rectángulos descritos por cuatro valores que representan las coordenadas superior-izquierda e inferior-derecha. Para el desarrollo del programa considere las siguientes definiciones:

```
typedef struct punto{
  int x;
  int y;
}punto;
```

```
typedef struct rectangulo{
    int x1,y1;
    int x2,y2;
}rectangulo;
```

Para cada punto la salida del programa debe ser:
Punto i esta contenido en el rectángulo j.
En caso que un punto no se encuentre contenido en ninguna figura, el mensaje deberá ser:
Punto i no esta contenido en ningún rectángulo.
Puntos y rectángulos se numeran en el orden en el cual aparecen en los datos de entrada. Además, considere que adicionalmente se entregan como datos de entrada el número de puntos y rectángulos.
Ejemplo

```
5
5 1
2 4
6 3
2 2
2 5
3
1 2 3 1
4 4 8 2
1 6 3 3
```

```
Salida: Punto 1 no esta contenido en ningun rectangulo.
        Punto 2 esta contenido en el rectangulo 3.
        Punto 3 esta contenido en el rectangulo 2.
        Punto 4 esta contenido en el rectangulo 1.
        Punto 5 esta contenido en el rectangulo 3.
```

2. En un archivo de texto, llamado "circunferencias.txt" se encuentra las coordenadas del centro (x, y) y radio r de circunferencias (una por línea). En otro archivo de texto llamado "puntos.txt" se encuentra las coordenadas

(x, y) de puntos. Considere que la primera línea de cada archivo contiene la cantidad de puntos y circunferencias. Escriba las siguientes funciones:

- **circunferencia * creaCircunf(void);** la cual crea un arreglo a partir de las datos almacenados en "circunferencias.txt".
- **punto * creaPuntos(void);** la cual crea un arreglo a partir de las datos almacenados en "puntos.txt".
- **void pertenece(punto *, circunferencia*);** la cual muestra por pantalla si un punto se encuentra al interior de alguna circunferencia. La salida, estará compuesta para cada punto, con un mensaje de la forma:
  Punto i **esta contenido en la figura** j.
  Si el punto no se encuentra contenido en ninguna figura el mensaje deberá ser:
  Punto i **no esta contenido en ninguna figura**.
  Puntos y figuras deben ser numeradas en el orden en el cual aparecen en sus respectivos archivos de entrada.

Para el desarrollo del ejercicio considere las sgtes definiciones:

```
 typedef struct punto{
 int x;
 int y;
}punto;

typedef struct circunferencia{
 int x;
 int y;
 int r;
}circunferencia;
```

3. Se tiene el siguiente problema matemático de variables enteras consistente en maximizar la función de $R^5 : f(x_1, x_2, x_3, x_4, x_5) = 2x_1 + x_2 - x_3 + 2x_4 + x_5$. Se poseen n soluciones almacenadas en un archivo de texto llamado *soluciones.txt*, de la siguiente forma:

```
4
3 3 3 1 6
4 1 4 2 3
2 2 5 2 5
1 1 1 1 1
```

En el ejemplo, el valor 4 indica la cantidad de soluciones almacenadas y las 4 filas que siguen representan una solución. Escriba la función **solucion *crearArreglo(void);** que almacena las variables x_1, x_2, x_3, x_4, x_5 y el valor de $f(x_1, x_2, x_3, x_4, x_5)$ en un arreglo de estructuras considerando la siguiente definición:

```
typedef struct solucion{
```

```
        int x1,x2,x3,x4,x5;
        int valor;
}solucion;
```

y además escriba la función **void mejorSolucion(solucion *);** que recibe como parámetro el arreglo creado en la función anterior y muestra cual es la solución que maximiza la función en el siguiente formato:

```
f(x1,x2,x3,x4,x5)=valor
```

para el ejemplo de más arriba la salida es:

```
f(3,3,3,1,6)=14
```

4. En un tablero de N filas y M columnas se encuentran todos los números entre 1 y $N \times M$. Con el objetivo de ordenar el contenido del tablero, en orden ascendente por fila, el único movimiento permitido es invertir el contenido de una fila o de una columna. Escriba un programa en C basado en funciones y uso de arreglos bidimensionales dinámicos que verifique si un tablero se encuentra ordenado luego de realizar una cierta cantidad de movimientos de fila y/o columna. En la figura se tiene un tablero de 4×4 el cual queda totalmente ordenado luego de realizar movimientos sobre la tercera fila (F3), la tercera columna (C3) y la segunda fila (F2).

```
1  2  15  4     F3     1  2  15  4     C3     1  2  3  4     F2     1  2  3  4
8  7  11  5    --->    8  7  11  5    --->    8  7  6  5    --->    5  6  7  8
12 6  10  9            9  10 6   12          9  10 11 12          9  10 11 12
13 14 3   16           13 14 3   16          13 14 15 16         13 14 15 16
```

Datos de entrada: los datos del tablero se encuentran en un archivo de texto llamado *tablero.txt*, la primera línea contiene N y M separados por un espacio. Luego, la siguientes N líneas contienen M números cada una. Por último, se especifica una serie de movimientos especificados por un caracter $\in \{F, C\}$ seguido por el número de fila y/o columna sobre la cual se desea realizar un movimiento. Los movimientos terminan cuando se encuentra el par Z 0.
Salida: el programa debe mostrar el mensaje "Tablero ordenado" ó "Tablero desordenado" según corresponda.
Ejemplo:

```
Entrada          |     Salida
4 4              |      Tablero ordenado
1  2  15  4      |
8  7  11  5      |
12 6  10  9      |
13 14 3   16     |
F 3              |
C 3              |
F 2              |
Z 0              |
```

5. En un archivo de texto llamado *palabras.txt* se encuentran almacenadas palabras de acuerdo al siguiente formato:

```
5
paralelepipedo
angulo
recta
lado
vertice
```

En el ejemplo, el valor 5 indica la cantidad de palabras almacenadas y en las 5 filas que siguen se encuentra cada una de las palabras. Considerando lo anterior:

- (a) implemente la función **palabra *crearArreglo(void);** que lee cada una de las palabras almacenadas en el archivo *palabras.txt* y las almacena junto a su respectivo largo (cantidad de caracteres de la palabra) en un arreglo de estructuras considerando la siguiente definición:

```
typedef struct palabra{
     char palabra[20];
     int  largo;
}palabra;
```

- (b) implemente la función **void palabraMasCorta(palabra *);** que recibe como parámetro el arreglo creado en la función anterior y muestra cual es la palabra más corta con el sgte formato:

  ```
  La palabra más corta es: xxxxxx
  ```

  para el ejemplo de más arriba la salida es:

  ```
  La palabra más corta es: lado
  ```

  Considere que el largo máximo de una palabra es 20 y que la variable global

  ```
  int numPalabras;
  ```

  se utiliza para almacenar el tamaño del arreglo de estructuras creado en la función **palabra *crearArreglo(void);**

6. El Código Da Vinci es uno de los libros más leídos y controversiales de todos los tiempos. En este libro el escritor Dan Brown utiliza una técnica de encriptación muy interesante para mantener un mensaje secreto. En este problema, se le dará una serie de números tomados desde la serie de Fibonacci y un texto encriptado. Su tarea consiste en descifrar el texto usando la técnica de desencriptado que se describe a continuación. Vamos a seguir el ejemplo. Cualquier texto cifrado consta de dos líneas. La primera línea contiene algunos números extraídos desde la serie de Fibonacci. La segunda línea es el texto encriptado. Así, dado el siguiente texto cifrado :
   13 - 2 - 89 - 377 - 8 - 3 - 233 - 34 - 144 - 21 - 1
   OH, LAME SAINT!
   La salida será: ***THE MONA LISA***

- En este ejemplo, el primer número es el 13, que es el sexto término de la serie de ***Fibonacci***. Así que la primera letra en mayúsculas en el texto cifrado ***'O'*** va en el sexto lugar de la cadena de salida.
- El segundo número es el 2, que es el segundo término de la serie de ***Fibonacci*** y por lo tanto la letra ***'H'*** va en la segunda posición en la cadena de salida;
- El tercer número es el 89 que es el décimo término de la serie de ***Fibonacci***, de modo que la letra ***'L'***va en la décima posición en la cadena de salida.
- Este proceso continúa hasta que se procesa el último número y entonces obtenemos la cadena ***"THE MONA LISA"***.

Tenga en cuenta que :

- Sólo las letras mayúsculas del alfabeto inglés forman parte del mensaje; el resto de los carácteres son considerados simplemente como basura.
- Las posiciones no ocupadas en la cadena de salida corresponden a espacios en blanco.

Datos de entrada

Los datos de entrada se encuentran en un archivo de texto llamado *texto.txt*. El contenido del archivo comienza con una línea que consta de un solo número T. Donde T será la cantidad de textos cifrados que siguen. Cada texto cifrado esta compuesto por tres líneas.

- La primera línea contiene un entero positivo N, considere que $1 \leq N \leq 100$
- La segunda línea hay N números extraídos de la serie de Fibonacci , considere que $1 \leq N_i \leq 10^{11}$.
- La tercera línea contiene el texto cifrado (Considere que el largo máximo del texto son 100 caracteres).

Salida

Para cada texto cifrado, la salida contiene una sola línea que contiene el texto descifrado. Recuerde que el texto descifrado contendrá sólo letras mayúsculas.

Ejemplo

Dada la siguiente entrada:

```
2
11
13 2 89 377 8 3 233 34 144 21 1
OH, LAME SAINT!
15
34 21 13 144 1597 3 987 610 8 5 89 2 377 2584 1
O, DRACONIAN DEVIL!
```

La salida es:

```
THE MONA LISA
LEONARDO DA VINCI
```

7. Dada una lista de figuras (rectángulos y círculos) y una lista de puntos en el plano cartesiano, escriba un programa en lenguaje C que mediante el uso de structs y arreglos dinámicos determine para cada punto si se encuentra contenido en alguna figura.
Entrada y Salida
La entrada se encuentra en un archivo de texto llamado **figuras.txt** el cual contendrá la descripción, una por línea, de **n** figuras ($n \leq 100$). El primer caracter representará el tipo de figura('r','c' para rectángulo y círculo respectivamente), el cual es seguido por valores que describen la figura:

   - Para un rectángulo, serán 4 valores reales correspondientes a las coordenadas x e y de las esquinas superior izquierda e inferior derecha.
   - Para un círculo, serán 3 valores, la coordenada x e y del centro y el radio.

   El fin de la lista de figuras será marcado mediante el signo +. Las líneas restantes contendrán las coordenadas, una por línea, de **m** los puntos a ser testeados, $m \leq 10$. El fin de esta lista será indicado mediante un punto de coordenadas 9999.99999.9 el cual no se considera dentro de la evaluación. Los puntos ubicados en el borde de una figura no se consideran como internos a ella.
   La salida, estará compuesta para cada punto, con un mensaje de la forma: **Punto** i **esta contenido en la figura** j.
   Si el punto no se encuentra contenido en alguna figura el mensaje deberá ser: **Punto** i **no esta contenido en alguna figura** j.
   Puntos y figuras deben ser numeradas en el orden en el cual aparecen en el archivo de entrada.
   Ejemplo:
   ENTRADA
   r 8.5 17.0 25.5 -8.5
   c 20.2 7.3 5.8
   r 0.0 10.3 5.5 0.0
   c -5.0 -5.0 3.7
   r 2.5 12.5 12.5 2.5
   c 5.0 15.0 7.2
   +

2.0 2.0
4.7 5.3
6.9 11.2
20.0 20.0
17.6 3.2
-5.2 -7.8
9999.9 9999.9

SALIDA

Punto 1 esta contenido en la figura 3
Punto 2 esta contenido en la figura 3
Punto 2 esta contenido in figura 5
Punto 3 esta contenido in figura 5
Punto 3 esta contenido in figura 6
Punto 4 no esta contenido en alguna figura
Punto 5 esta contenido in figura 1
Punto 5 esta contenido in figura 2
Punto 6 esta contenido in figura 4

8. En un archivo de texto llamado *binario.txt* se encuentran almacenados números binarios de la sgte manera:

```
10101000
1110
1000111110
1011001
....
```

Escriba un programa que implemente la función

int convertirADecimal(char *); la cual recibe un nro en binario y entrega su equivalente en decimal.

La salida del programa debe mostrar para cada número binario su equivalente en decimal de la sgte manera:

```
El nro 111 (base 2) = 7 (base 10)
```