# Clase 02

## 1 Bubble Sort (Extraído de Lab3 – 2014-2)

Print the partially sorted arrays that result in each iteration of bubble sort. For each test case the first digit represents the number of elements to be sorted.

| Sample Input | Sample Output |
|---|---|
| 9 12 3 100 6 43 18 8 12 2<br>4 200 60 20 8<br>4 -200 -60 -20 -8 | 3 12 6 43 18 8 12 2 100<br>3 6 12 18 8 12 2 43 100<br>3 6 12 8 12 2 18 43 100<br>3 6 8 12 2 12 18 43 100<br>3 6 8 2 12 12 18 43 100<br>3 6 2 8 12 12 18 43 100<br>3 2 6 8 12 12 18 43 100<br>2 3 6 8 12 12 18 43 100<br>60 20 8 200<br>20 8 60 200<br>8 20 60 200<br>-200 -60 -20 -8<br>-200 -60 -20 -8<br>-200 -60 -20 -8 |

- **EXTRA:** ¿Cómo mejoraría el algoritmo para evitar que siga recorriendo la lista de elementos cuando ya está ordenada?

## 2 String Matching

Modifique su implementación de algoritmo **brute force string matching** para que ahora indique cuántas veces se repite el *patrón* en el *texto*.

## 3 C Programming

**a)** Indique si las siguientes secciones de código funcionan o no. En caso no lo hagan indique por qué y cómo corregiría el error.

```
int main() {
     x = 10;
     printf (“x = %d\n”, x);
}
```

```
char x = 'Y';
while (x = 'Y') {
  //...
  printf("Continue? (Y/N)");
  scanf(“%c”, &x);
}
```

```
int array[10];
//...
for(int x=1; x<=10; x++)
     printf(“%d ”, array[x]);
```

```
int main(){
     int x, y;
     scanf(“%d %d”, x, y);
}
```

```
char texto[11] = “Hola mundo”;
printf(“%s”, texto);
```

```
char st1[] = "abc";
char st2[] = "abc";
if (st1 == st2)
      printf("Yes");
else
      printf("No");
```

```
int x;
char * st = malloc(31);

scanf("%d", x);
scanf("%30s", &st);
```

```
#include <string.h>
int main() {
  char * st;
  strcpy(st, "abc");
  return 0;
}
```

```
int main() {
  menu();
}
void menu() {
  //...
}
```

```
int n;
scanf(“%d”, &n);
int array[n];
```

```
// str es un variable char []
char *p = (char *)
malloc(strlen(str));
strcpy(p, str);
```

```
int * ptr , m = 100 ;
ptr = m ;
```

```
int * ptr , m = 100 ;
*ptr = m ;
```

```
int * ptr , m = 100 ;
ptr =  &m ;
printf("%d",ptr);
```

# 4 UVa 10662 - The Wedding (Traducción Libre)

Félix y Leti se van a casar pronto y todos están preparando sus regalos de boda, pero tienen un serio problema: el dinero; su presupuesto no es muy alto. A ellos les gustaría reservar un buen restaurante, "dormir" su primera noche en un buen hotel y pasar una maravillosa luna de miel viajando alrededor del mundo. La mejor manera de obtener el precio más barato es adquirir un paquete todo incluido; es decir, se tiene que contratar el viaje, el restaurante y el hotel todo junto. ¿Es eso posible?

El problema consiste en encontrar la combinación de agencia de viaje - restaurante - hotel más barata. La "dificultad" radica en que no todas las combinaciones son posibles.

**Entrada:** Cada caso de prueba tiene el siguiente formato:

- La primera línea consiste de 3 números enteros T, R, H que indican el número de agencias de viaje, restaurantes y hoteles, respectivamente. Asuma que T < 20, R < 20 y H < 20. Las agencias de viaje, restaurantes y hoteles están numerados: 0, 1, 2, ...
- Las siguientes T + R + H líneas están divididas en tres bloques:
  - El primer bloque tiene T filas y R + 1 columnas. La primera columna corresponde a los precios ofrecidos por las agencias de viajes para el viaje alrededor del mundo. En el resto de columnas, la celda (i,j) es 0 si la agencia de viajes i se puede combinar con el restaurante j y 1 si no es posible.
  - El segundo bloque tiene R filas y H +1 columnas. La primera columna corresponde a los precios ofrecidos por los restaurantes. En el resto de columnas, la celda (i, j) es 0 si el restaurante i se puede combinar con el hotel j y 1 si no es posible.
  - El tercer bloque tiene H filas y T + 1 columnas. La primera columna corresponde a los precios ofrecidos por los hoteles. En el resto de columnas, la celda (i,j) es 0 si el hotel i se puede combinar con la agencia de viajes j y 1 si no es posible.
- La entrada culmina con una línea en blanco.

**Salida:** Para cada caso de prueba, la salida debe consistir de una sola línea con el número de agencia de viajes (T), restaurante (R) y hotel (H), y el precio total más barato (P). Estos valores deben ser escritos en el formato: T R H: P. Si no hay combinación posible, la salida debe ser una línea con el texto: Don't get married!. Si existe más de una posibilidad, puede escribir cualquiera.

| Ejemplo de Entrada | Ejemplo de Salida |
|---|---|
| 2 2 2<br>12 0 0<br>1 1 1<br>34 0 0<br>3 1 1<br>21 1 0<br>2 1 0<br>2 2 2<br>12 0 0<br>1 0 0<br>34 0 0<br>3 0 0<br>21 0 0<br>2 0 0 | Don't get married!<br>1 1 1:6 |

# 5 UVa 156 – Ananagrams

Most crossword puzzle fans are used to anagrams – groups of words with the same letters in different orders – for example OPTS, SPOT, STOP, POTS and POST. Some words, however, do not have this attribute; no matter how you rearrange their letters, you cannot form another word. Such words are called ananagrams; an example is QUIZ.

Obviously, such definitions depend on the domain within which we are working. You might think that ATHENE is an ananagram, whereas any chemist would quickly produce ETHANE. One possible domain would be the entire English language, but this could lead to some problems. One could restrict the domain to, say, Music, in which case SCALE becomes a relative ananagram (LACES is not in the same domain) but NOTE is not since it can produce TONE.

Write a program that will read in the dictionary of a restricted domain and determine the relative ananagrams. Note that single letter words are, ipso facto, relative ananagrams since they cannot be “rearranged” at all. The dictionary will contain no more than 1000 words.

**Input:** Input will consist of a series of lines, each with one word. Words consist of up to 20 upper and/or lower case letters. Note that words that contain the same letters but of differing case are considered to be anagrams of each other, thus tIeD and EdiT are anagrams. The file will be terminated by a line consisting of a single #.

**Output:** Output will consist of a series of lines. Each line will consist of a single word that is a relative ananagram in the input dictionary. Words must be output in lexicographic (case-sensitive) order. There will always be at least one relative ananagram.

| Sample Input | Sample Output |
|---|---|
| ladder<br>came<br>tape<br>soon<br>leader<br>acme<br>RIDE<br>lone<br>Dreis<br>peat<br>ScAlE<br>orb<br>eye<br>Rides<br>dealer<br>NotE<br>derail<br>LaCeS<br>drIed<br>noel<br>dire<br>Disk<br>mace<br>Rob<br>dries<br># | Disk<br>NotE<br>derail<br>drIed<br>eye<br>ladder<br>soon |