PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA
Primer Examen
(Primer Semestre de 2014)

Horario 0581: prof. Andrés Melgar
Horario 0582: prof. Fernando Alva

Duración: 3 horas
Nota:

- No se permite el uso de material de consulta.
- No se ofrecerá asesoría en la parte teórica.
- **Debe utilizar comentarios para explicar la lógica seguida en los programas elaborados, así como nombres de variables apropiados.**
- La presentación, la ortografía y la gramática influirán en la calificación.

Puntaje total: 20 puntos

Cuestionario:

PARTE TEÓRICA

Responda las siguientes preguntas según los conceptos vistos en clase:

Pregunta 1 (1 punto) ¿Por qué se dice que los arreglos, las matrices y las cadenas de caracteres se pasan siempre por referencia? **Justifique adecuadamente su respuesta**.

Pregunta 2 (0.5 puntos) ¿Qué almacena un puntero? ¿Para qué sirve un puntero? **Justifique su respuesta con un ejemplo**.

Pregunta 3 (0.5 puntos) Indique tres formas diferentes de inicializar un puntero. **Explique cada una de ellas**.

Pregunta 4 (1 punto) Ever, un alumno de TP se encuentra conversando con Katerine, una alumna de Algoritmia. Ever le explica a Katerine que desea codificar un programa para calcular un valor de la serie de fibonacci. Ever además le dice que ha leído sobre recursión y que no sabe si lo mejor es implementar el programa usando solamente estructuras iterativas o recursivas. Katerine, le comenta que en este caso lo mejor es usar la estrategia de "Programación Dinámica". ¿Qué opina Ud. sobre lo dicho por la alumna Katerine? ¿Concuerda con la afirmación? **Justifique adecuadamente su respuesta**.

Pregunta 5 (2 puntos) El siguiente programa implementa el algoritmo de ordenación *Several Unique*. Basado en el algoritmo que subyace al programa, se le pide de indique cuál es la estrategia usada en el algoritmo de ordenación. **Justifique adecuadamente su respuesta. Recuerde que no se está solicitando el pseudocódigo**.

Programa 1: Obtenido a través de la URL `http://leehaywood.org/misc/several-unique/`

```c
#include <limits.h>

    /* ... */

    int Element[ NUM_ELEMENTS ]; /* Array of signed integers to be sorted. */
    int EndIndex;
    int Position;
    int HighValue;
    int SwapIndex;
    int NewValue;

    EndIndex = NUM_ELEMENTS - 1;
    do
    {
        HighValue = INT_MIN;
        Position = -1;
        while ( Position < EndIndex )
        {
            Position++;
            NewValue = Element[ Position ];
            if ( NewValue < HighValue )
            {
                Element[ SwapIndex ] = NewValue;
                SwapIndex++;
                Element[ Position ] = HighValue;
            }

            if ( NewValue > HighValue )
            {
                SwapIndex = Position;
                HighValue = Element[ Position ];
            }
        }
        EndIndex = SwapIndex - 1;
    }
    while ( Position >= SwapIndex );
```

PARTE PRÁCTICA

Pregunta 6 (5 puntos) Se tiene un archivo de texto en donde en cada línea se encuentra: i) el código del alumno (número entero *e. g.* 19941146), ii) el código del curso matriculado (cadena de caracteres *e. g.* INF144), iii) el año (*e. g.* 2014), iv) el semestre (*e. g.* 1) y v) la nota obtenida (*e. g.* 16.7). Se desea obtener una serie de reportes como:

- Listado de alumnos que han llevado un determinado curso por tres o más veces
- Promedios obtenidos por curso por año
- Promedio de notas historico por curso

Se le pide a Ud. que implemente los reportes sabiendo que en el archivo se encuentra los datos de todos los alumnos que han cursado algun curso en la Faculdad de Ciencias e Ingeniería desde su creación.

a) (1.5 puntos) Describa la estrategia que usará

b) (3.5 puntos) Implemente la estrategia usando ANSI C.

Pregunta 7 (4 puntos) Un archivo de texto contiene todos los números enteros entre 1 y 10,000 (inclusive y sin repeticiones) de forma desordenada. El entero en la i-ésima fila del archivo corresponde a la i-ésima entrada en un arreglo.

Su tarea es implementar un programa en C que permita calcular el número total de comparaciones usadas para ordenar los número del archivo de entrada dado usando QuickSort. Como sabe, el número de comparaciones depende de cuáles elementos son seleccionados como pivotes, así que se desea que el programa sea lo suficientemente flexible para experimentar con diferentes reglas para seleccionarlos:

- Usar el primer elemento del arreglo como elemento pivote.
- Usar el último elemento del arreglo como elemento pivote.
- Usar la regla de la "mediana de tres".

Para el caso de la última regla, considere el primer elemento, el elemento del medio y el último elemento del arreglo dado. Si el arreglo tiene longitud impar queda claro cuál es el elemento del medio, pero si tiene longitud par 2k, use el k-ésimo elemento como elemento del medio. Por ejemplo, para el arreglo [4 5 6 7], el elemento del medio es el segundo – ¡5 y no 6!. Finalmente, identifique cuál de estos tres elementos es la mediana (i.e., aquel cuyo valor se encuentra entre los otros dos) y úselo como pivote.

Por ejemplo, si el arreglo de entrada fuese [8 2 4 5 7 1], debería considerar el primer elemento (8), el del medio (4), y el último (1). Como 4 es la mediana del conjunto {1,4,8}, debe usarlo como pivote.

PARTE ELECTIVA

Para **UNA** de las preguntas que se presentan a continuación, elabore un programa en C que resuelva el problema descrito.

Pregunta 8 (6 puntos) UVa 662 - Fast Food (Traducción Libre) La cadena de restaurantes de comida rápida McBurger posee diferentes restaurantes a lo largo de la carretera. Recientemente, han decidido construir varios almacenes a lo largo de la carretera, cada uno ubicado en un restaurante y encargado de abastecer a varios de los restaurantes de los ingredientes necesarios. Naturalmente, estos almacenes deben ser ubicados de tal forma que la distancia promedio entre un restaurante y su almacén designado sea la mínima posible. Se le pide escribir un programa que determine las posiciones óptimas y las asignaciones de los almacenes.

Para ser más precisos, la administración de McBurger ha emitido las siguientes especificaciones: se le proporcionará las posiciones de n restaurantes a lo largo de la carretera como n números enteros $d_1 < d_2 < ... < d_n$ (estas son las distancias medidas desde la sede de la empresa que se encuentra en la misma carretera). Además, un número k ($k < n$) indicará el número de almacenes que serán construidos.

Los k almacenes serán construidos en las ubicaciones de k restaurantes distintos. A cada restaurante se le asignará el almacén más cercano, desde el cuál recibirá sus suministros. Para minimizar los costos de envío, la *suma total de la distancia*, definida como

$$\sum_{i=1}^{n} |d_i - (\text{posición del almacén que atiende al restaurante i})|$$

debe ser lo más pequeña posible.

Entrada: La entrada contiene varias descripciones de cadenas de restaurantes de comida rápida. Cada descripción comienza con una línea que contiene dos números enteros n y k. Ambos satisfacen $1 \leq n \leq 200$, $1 \leq k \leq 30$, $k \leq n$. A continuación, aparecen n líneas que contienen un número entero cada una que indican las posiciones d_i de los restaurantes, ordenadas de forma creciente. La entrada culminará con un caso que comienza con $n = k = 0$. Este caso no debe ser procesado.

Salida: Para cada cadena, primero imprima el número de la cadena. Luego, imprima la ubicación óptima de los almacenes com sigue: para cada almacén imprima una linea que contenga su posición y el rallo de restaurantes que atiende. Si hay más de una solución óptima, imprima cualquiera de ellas. Después de la descripción del almacén, imprima una línea que contenga la suma total de la distancia, tal y como se definió previamente. Imprima una línea en blanco después de cada caso de prueba.

Ejemplo de Entrada:

```
6 3
5
6
12
19
20
27
0 0
```

Ejemplo de Salida:

```
Cadena 1
Almacén 1 en el restaurante 2 atiende restaurantes 1 a 3
Almacén 2 en el restaurante 4 atiende restaurantes 4 a 5
Almacén 3 en el restaurante 6 atiende restaurante 6
Suma total de la distancia = 8
```

Pregunta 9 (6 puntos) UVa 604 - The Boggle Game (Traducción Libre) El idioma PigEwu tiene una sintaxis bastante simple. Cada palabra en este idioma tiene exactamente 4 letras. Además, cada palabra contiene exactamente 2 vocales ("y" es considerada una vocal en PigEwu). Por ejemplo, "ama" y "eve" son palabras legítimas, "arts" no es una palabra válida.

En el juego boggle, se le da un tablero de letras de 4x4 y se le pide encontrar todas las palabras contenidas en el tablero. Una palabra en nuestro caso (PigEwu) será una secuencia de 4 cuadrados distintos (letras) que formen una palabra legítima, tal que cada cuadrado toque (i.e., tenga una esquina o un lado en común) el siguiente cuadrado. Por ejemplo:

```
A  S  S  D
S  B  E  Y
G  F  O  I
H  U  U  K
```

En este tablero, una lista parcial de palabras válidas incluye:

```
ASGU    SABO    FOIK    FOYD    SYDE    HUFO
```

BEBO es una palabra válida pero no lo es en este tablero de boggle porque no hay dos B's.

Escriba un programa que lea un par de tableros de boggle y liste todas las palabras PigEwu comunes a ambos tableros.

Entrada: La entrada contiene varios conjuntos de prueba. Cada conjunto de prueba consistirá de un par de tableros como se muestra en el Ejemplo de Entrada. Todas las letras estarán en mayúsculas. Dos letras consecutivas en el mismo tablero estarán separadas por un espacio en blanco. La primera fila del primer tablero estará en la misma línea que la primera fila del segundo tablero. La separación entre ellos será de 4 espacios, y lo mismo se mantendrá para las 3 filas restantes. Los pares de tableros estarán separados por una línea en blanco. La entrada culminará con un #.

Salida: Para cada par de tableros de boggle, imprima un lista ordenada alfabéticamente de todas las palabras comunes, cada palabra en una línea diferente; o el enunciado "No existen palabras comunes para este par de tableros de boggle". Separe la salida de cada par de tableros de boggle con una línea en blanco.

Ejemplo de Entrada:

```
D F F B    W A S U
T U G I    B R E T
O K J M    Y A P Q
```

```
K M B E    L O Y R

Z W A V    G S F U
U N C O    A H F T
Y T G I    G N A L
H G P M    B O O B

#
```

Ejemplo de Salida:

```
No existen palabras comunes para este par de tableros de boggle.

ANGO
AOGN
GNAO
GOAN
NAOG
NGOA
OANG
OGNA
```

Profesores del curso: Andrés Melgar
Fernando Alva

Pando, 17 de mayo de 2014