**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ**
**FACULTAD DE CIENCIAS E INGENIERÍA**

# ALGORITMIA
**Examen 2**
**(Primer semestre 2015)**

**Indicaciones generales:**

- Duración: 2h 55 min.
- Materiales o equipos a utilizar: No se permite el uso de material de consulta.
- Al inicio de cada programa, el alumno deberá incluir, a modo de comentario, la estrategia que utilizará para resolver el problema. De no incluirse dicho comentario o si la implementación es significativamente diferente a la estrategia indicada, el alumno no podrá obtener el puntaje completo en dicha pregunta.
- Un programa que no muestre resultados coherentes y/o útiles será corregido sobre el 60 % del puntaje asignado a dicha pregunta.
- **Debe utilizar comentarios para explicar la lógica seguida en el programa elaborado.**
- Cada programa debe ser guardado en un archivo con el nombre *preg#codigo_de_alumno.c* y subido a PAIDEIA en el espacio indicado por los Jefes de Práctica.

Puntaje total: 20 puntos

---

Cuestionario:

## Pregunta 1 (6 puntos) Aplicación de TADs

Indique el nombre o número de equipo en el que realizó el segundo trabajo grupal.

## Pregunta 2 (5 puntos) Hashing

Sea $T$ una tabla de hash de tamaño 5 y $h$ la siguiente función de hash: $h(k) = 4 + 3k \bmod 5$, donde $k = \sum$ *posiciones en el alfabeto de las letras que componen la palabra*

Se desea insertar en $T$ los nombres de los países que participan en la Copa América Chile 2015: PERU, CHILE, BRASIL, ARGENTINA, BOLIVIA, COLOMBIA, ECUADOR, URUGUAY, PARAGUAY, VENEZUELA, JAMAICA y MEXICO en ese mismo orden usando $h$.

| A | B | C | D | **E** | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |

Implemente un programa en C que permita:

a) Mostrar el estado de la tabla hash luego de insertar claves en $T$ si las colisiones se resuelven por "Open Hashing".

b) Mostrar el estado de la tabla hash luego de insertar claves en $T$ si las colisiones se resuelven por "Closed Hashing".

**NOTA:** Deberá implementar todos los TADs que requiera para manipular las Tablas Hash con su respectivo manejo de colisiones.

## Pregunta 3 (5 puntos) UVa 10986 - Sending email

There are $n$ SMTP servers connected by network cables. Each of the $m$ cables connects two computers and has a certain latency measured in milliseconds required to send an email message. What is the shortest time required to send a message from server $S$ to server $T$ along a sequence of cables? Assume that there is no delay incurred at any of the servers.

**Input:** The first line of input gives the number of cases, $N$. $N$ test cases follow. Each one starts with a line containing $n$ ($2 <= n < 20000$), $m$ ($0 <= m < 50000$), $S$ ($0 <= S < n$) and $T$ ($0 <= T < n$). $S! = T$. The next $m$ lines will each contain 3 integers: 2 different servers (in the range $[0, n-1]$) that are connected by a bidirectional cable and the latency, $w$, along this cable ($0 <= w <= 10000$).

**Output:** For each test case, output the line "Case #x" followed by the number of milliseconds required to send a message from $S$ to $T$. Print "unreachable" if there is no route from $S$ to $T$.

| **Sample Input:** | **Sample Output:** |
|---|---|
| 3<br>2 1 0 1<br>0 1 100<br>3 3 2 0<br>0 1 100<br>0 2 200<br>1 2 50<br>2 0 0 1 | Case #1: 100<br>Case #2: 150<br>Case #3: unreachable |

Implemente un programa en C que resuelva el problema descrito.

# PARTE ELECTIVA

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

## Pregunta 4 (4 puntos) UVa 127 - "Accordian" Patience

You are to simulate the playing of games of "Accordian" patience, the rules for which are as follows:

Deal cards one by one in a row from left to right, not overlapping. Whenever the card matches its immediate neighbour on the left, or matches the third card to the left, it may be moved onto that card. Cards match if they are of the same suit or same rank. After making a move, look to see if it has made additional moves possible. Only the top card of each pile may be moved at any given time. Gaps between piles should be closed up as soon as they appear by moving all piles on the right of the gap one position to the left. Deal out the whole pack, combining cards towards the left whenever possible. The game is won if the pack is reduced to a single pile.

Situations can arise where more than one play is possible. Where two cards may be moved, you should adopt the strategy of always moving the leftmost card possible. Where a card may be moved either one position to the left or three positions to the left, move it three positions.

**Input:** Input data to the program specifies the order in which cards are dealt from the pack. The input contains pairs of lines, each line containing 26 cards separated by single space characters. The final line of the input file contains a # as its first character. Cards are represented as a two character code. The first character is the face-value (A=Ace, 2-9, T=10, J=Jack, Q=Queen, K=King) and the second character is the suit (C=Clubs, D=Diamonds, H=Hearts, S=Spades).

**Output:** One line of output must be produced for each pair of lines (that between them describe a

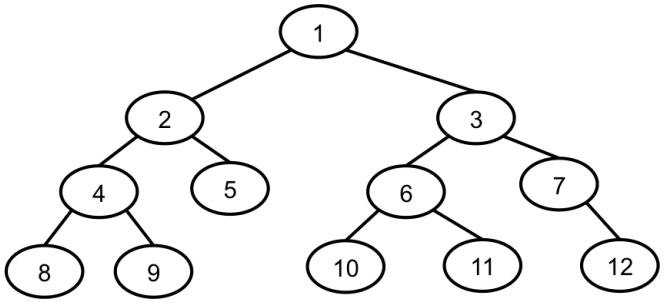
pack of 52 cards) in the input. Each line of output shows the number of cards in each of the piles remaining after playing "Accordian patience" with the pack of cards as described by the corresponding pairs of input lines.

**Sample Input:**
QD AD 8H 5S 3H 5H TC 4D JH KS 6H 8S JS AC AS 8D 2H QS TS 3S AH 4H TH TD 3C 6S
8C 7D 4C 4S 7S 9H 7C 5D 2S KD 2D QH JD 6D 9D JC 2C KH 3D QC 6C 9S KC 7H 9C 5C
AC 2C 3C 4C 5C 6C 7C 8C 9C TC JC QC KC AD 2D 3D 4D 5D 6D 7D 8D TD 9D JD QD KD
AH 2H 3H 4H 5H 6H 7H 8H 9H KH 6S QH TH AS 2S 3S 4S 5S JH 7S 8S 9S TS JS QS KS
#

**Sample Output:**
6 piles remaining: 40 8 1 1 1 1
1 pile remaining: 52

## Pregunta 5 (4 puntos) Perímetro de Árbol

Implemente una función que reciba como parámetro un árbol binario e imprima su perímetro. Por ejemplo, dado el árbol:




el perímetro a imprimir es: 1 - 2 - 4 - 8 - 9 - 5 - 10 - 11 - 12 - 7 - 3

**NOTA:** Deberá implementar todos los TADs necesarios, así como la función main que verifique el correcto funcionamiento de la función solicitada.

Profesores del curso: Fernando Alva
Robert Ormeño

Pando, 04 de julio de 2015