UNIVERSIDAD DE COSTA RICA

Escuela de Ciencias de la Computación e Informática

CI-1221 Estructuras de Datos y Análisis de Algoritmos
I ciclo 2014, grupo 2

I Examen Parcial

Martes 29 de abril

Nombre: _____ Carné: _____

El examen consta de 9 preguntas que suman 120½ puntos, pero no se reconocerán más de 110 (10 % extra). Cada pregunta indica el tema tratado y su valor. Si la pregunta tiene subítemes, el puntaje de cada uno es indicado en los subítemes. Se recomienda echar un vistazo a los temas de las preguntas y a su puntaje antes de resolver el examen, para así distribuir su tiempo y esfuerzo de la mejor manera.

Las preguntas se pueden responder en cualquier orden, pero se debe indicar en la tabla mostrada abajo los números de página del cuaderno de examen en la que están las respuestas. Para esto debe numerar las hojas del cuaderno de examen en la esquina superior externa de cada página. Si la respuesta está en el enunciado del examen favor indicarlo con la letra *E* en vez del número de página. El examen se puede realizar con lápiz o lapicero. *No se permite el uso de dispositivos electrónicos (calculadoras, teléfonos, audífonos, etc.).*

Pregunta	Puntos	Páginas	Calificación
1. *Algoritmos iterativos*	45		
2. *Algoritmos recursivos*	10		
3. *Solución de recurrencias*	20		
4. *Ordenamiento por selección*	6		
5. *Ordenamiento por inserción*	6		
6. *Ordenamiento por mezcla*	6		
7. *Ordenamiento por montículos*	9		
8. *Ordenamiento rápido*	6		
9. *Ordenamiento por residuos*	12½		
Total	120½		

Las preguntas 4 a 9 se refieren a los últimos siete presidentes de nuestro país, incluyendo al recientemente electo: Sr. Luis Guillermo Solís Rivera.

1. *Algoritmos iterativos.* [45 pts.]
La criba de Eratóstenes es un algoritmo para encontrar los números primos que no exceden un n dado. A continuación se muestra en seudocódigo una versión simplificada del algoritmo:

Procedimento Criba-de-Eratóstenes(n)
1 **para** $i = 2, 3, \ldots, n/2$
2 marque los múltiplos de i, excepto i $(2i, 3i, 4i, \ldots)$, como compuestos
3 **despliegue** los no compuestos, excepto la unidad.[a]

---

[a] La unidad no es primo ni compuesto.

a) Escriba un algoritmo en C o C++ que despliegue en la salida estándar los números primos que no excedan n. Utilice el siguiente encabezado: [12 pts.]

```
void criba( int n );
```

b) Establezca la correctitud del algoritmo identificando un invariante apropiado para el ciclo principal del algoritmo [3 pts.] y mostrando cómo los pasos de inicialización [3 pts.], mantenimiento [3 pts] y terminación [2 pts.] permiten determinar la correctitud del algoritmo.

c) Asumiendo que una asignación toma un tiempo t_a, una comparación t_c, un incremento o decremento por la unidad ($++$ o $--$) t_i, una suma t_s y una multiplicación t_m, escriba una fórmula para el tiempo de ejecución del algoritmo. [12 pts.]

d) Determine una cota asintótica lo más precisa posible para el tiempo de ejecución del algoritmo. [10 pts.]

2. *Algoritmos recursivos.* [10 pts.]
El juego de las torres de Hanoi consiste en mover n discos de un estaca a otra utilizando como pivote una tercera estaca. Los discos se mueven uno a la vez, sin colocar discos encima de discos más pequeños. Se empieza con los discos apilados del más grande al más pequeño en una de las estacas, como se muestra a continuación para $n = 4$:

El problema se puede resolver siguiendo las instrucciones desplegadas por este algoritmo:

Procedimento Hanoi(n, aquí, allá, pivote)
1 **si** $n = 1$
2 **despliegue** "Mover disco de " aquí " a " allá
3 **sino**
4 Hanoi($n - 1$, aquí, pivote, allá)
5 **despliegue** "Mover disco de " aquí " a " allá
6 Hanoi($n - 1$, pivote, allá, aquí)

---

a) Escriba una fórmula recursiva para el tiempo de ejecución del algoritmo. [3 pts.]

b) Determine una cota asintótica para el tiempo de ejecución del algoritmo. [7 pts.]

3. *Solución de recurrencias.* [20 pts.]
   Resuelva las siguientes recurrencias asumiendo que $T(n) = \Theta(1)$ para $n \leq 1$.
   a) $T(n) = T(n/2) + \lg n^2$. [14 pts.]
   b) $T(n) = 7\,T(n/2) + n^2$. [6 pts.]

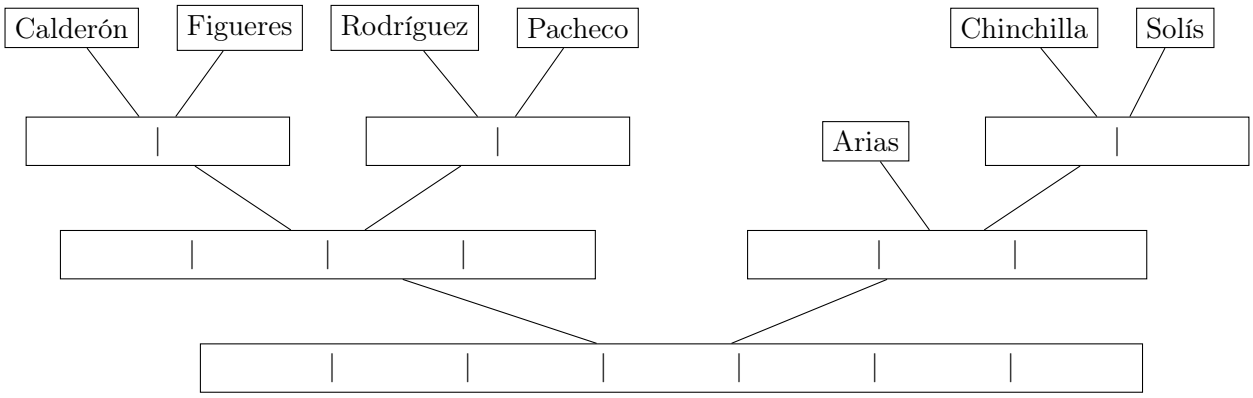
4. *Ordenamiento por selección.* [6 pts.]
   Simule la ejecución del algoritmo de ordenamiento por selección sobre el arreglo mostrado abajo. Muestre el estado del arreglo al finalizar cada una de las iteraciones del ciclo principal (externo). Si no muestra el estado de una casilla se asume que conserva el mismo valor que en la iteración anterior. Después de la primer inserción incorrecta el resto de inserciones no suman puntos.

	Posición						
Itn.	1	2	3	4	5	6	7
0	Calderón	Figueres	Rodríguez	Pacheco	Arias	Chinchilla	Solís
1							
2							
3							
4							
5							
6							

5. *Ordenamiento por inserción.* [6 pts.]
   Simule la ejecución del algoritmo de ordenamiento por inserción sobre el arreglo mostrado abajo. Muestre el estado del arreglo al finalizar cada una de las iteraciones del ciclo principal (externo). Si no muestra el estado de una casilla se asume que conserva el mismo valor que en la iteración anterior. Después de la primer inserción incorrecta el resto de inserciones no suman puntos.

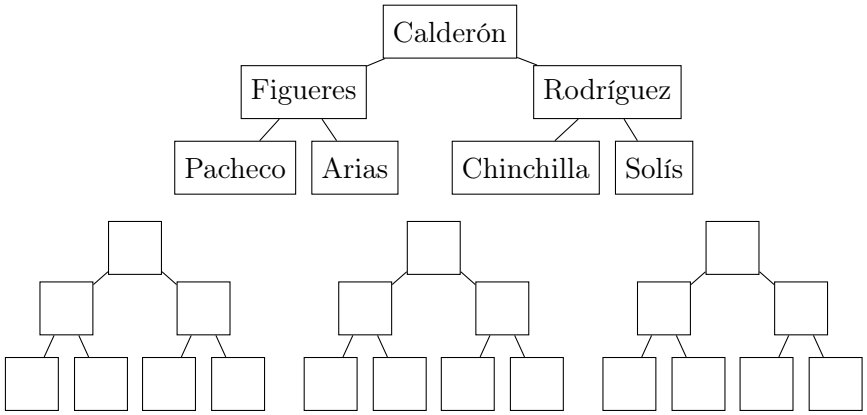
	Posición						
Itn.	1	2	3	4	5	6	7
1	Calderón	Figueres	Rodríguez	Pacheco	Arias	Chinchilla	Solís
2							
3							
4							
5							
6							
7							

6. *Ordenamiento por mezcla.* [6 pts.]
   Simule la ejecución del algoritmo de ordenamiento por mezcla sobre el arreglo mostrado abajo. Muestre el estado del arreglo al finalizar cada una de las mezclas. Si no muestra el estado de una casilla se asume que conserva el mismo valor que en la iteración anterior. Si el tamaño del (sub) arreglo es impar, *redondee* el punto medio hacia abajo, es decir, la parte izquierda del subarreglo debe ser más pequeña que la parte derecha. Por ejemplo, un arreglo de tamaño 3 debe ser partido en un arreglo de tamaño 1 (a la izquierda) y uno de tamaño 2 (a la derecha). Después de la primer mezcla incorrecta el resto de mezclas no suman puntos.

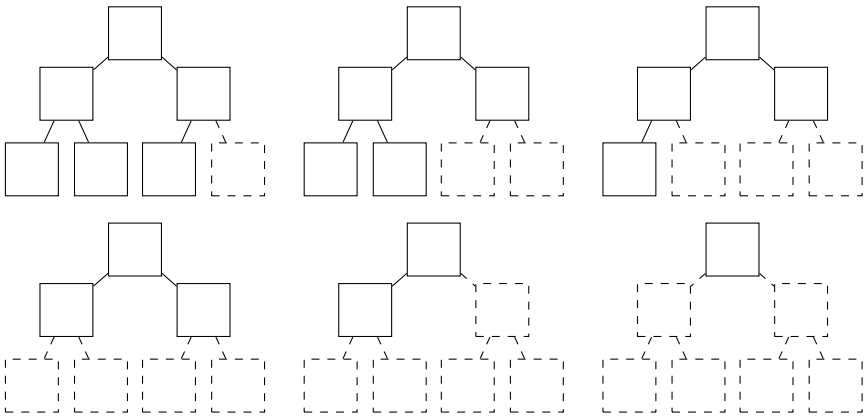


7. *Ordenamiento por montículos.* [9 pts.]
   Simule la ejecución del algoritmo de ordenamiento por montículos sobre el arreglo mostrado abajo en forma de árbol. Muestre el estado del *montículo* después de cada llamado a Corregir-Cima y Extraer-Máximo. (Los nodos con línea discontinua los puede dejar vacíos si lo desea). Después de la primer operación incorrecta el resto de operaciones no suman puntos.

   Monticularizar:



   Ordenar:



8. *Ordenamiento rápido.* [6 pts.]

Simule la ejecución del algoritmo de ordenamiento rápido sobre el arreglo mostrado abajo. Muestre el estado del subarreglo a ordenar después de cada llamado a PARTICIÓN e indique la posición del pivote devuelto. Si no muestra el estado de una casilla se asume que conserva el mismo valor que en la iteración anterior. Después del primer estado incorrecto el resto de estados no suman puntos.

Llamado	Estado del (sub)arreglo						
	1	2	3	4	5	6	7
	Calderón	Figueres	Rodríguez	Pacheco	Arias	Chinchilla	Solís
1.º	1	2	3	4	5	6	7
2.º	1	2	3	4	5	6	7
3.º	1	2	3	4	5	6	7
4.º	1	2	3	4	5	6	7
5.º	1	2	3	4	5	6	7
6.º	1	2	3	4	5	6	7

9. *Ordenamiento por residuos.* [12½ pts.]

Simule la ejecución del algoritmo de ordenamiento por residuos sobre las iniciales del primer nombre y apellido de los últimos siete presidentes que ha tenido nuestro país (incluyendo al recién electo Luis Solís). Muestre lo siguiente para cada llamado a la subrutina de ordenamiento por conteo:

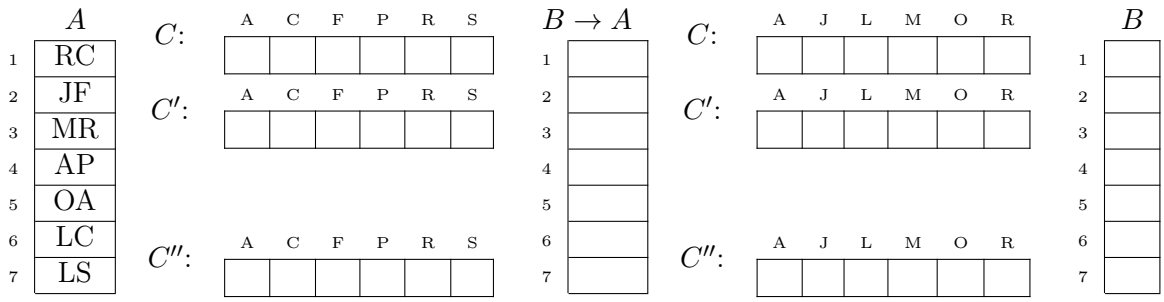
I. El histograma C. [3 pts.]

II. El histograma acumulativo C'. [3 pts.]

III. El arreglo resultante B. [3½ pts.]

IV. El estado final del histograma acumulativo C'' (después de producir el arreglo B). [3 pts.]

Después del primer arreglo incorrecto el resto de arreglos no suman puntos.


A
1 RC
2 JF
3 MR
4 AP
5 OA
6 LC
7 LS
C:
A C F P R S
C′:
A C F P R S
C″:
A C F P R S
B → A
1
2
3
4
5
6
7
C:
A J L M O R
C′:
A J L M O R
C″:
A J L M O R
B
1
2
3
4
5
6
7