UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

E.T.S.E

PROGRAMACIÓN II

Examen

Profesor: David E. Losada **Mayo 2011**

Fecha: 31/5/2011.
Nombre y apellidos:_____

PARTE I (SIN MATERIAL)

1. Test (3 puntos. Cada respuesta correcta suma 0.15 puntos. Cada respuesta incorrecta resta 0.07 puntos)

    1. La parte sintáctica de la especificación formal de un Tipo Abstracto de Dato (TAD)
        a. explicita los detalles de implementación de las operaciones del nuevo de tipo de datos
        b. explicita las operaciones (argumentos que tienen y valores que devuelven) del TAD
        c. explicita el comportamiento o semántica que tienen las operaciones del TAD
        d. ninguna de las anteriores

    2. La parte semántica de la especificación formal de un Tipo Abstracto de Dato (TAD)
        a. explicita los detalles de implementación de las operaciones del nuevo de tipo de datos
        b. explicita exclusivamente las operaciones (argumentos que tienen y valores que devuelven) del TAD
        c. describe de manera informal que tipo de argumentos se esperan en las operaciones del TAD
        d. ninguna de las anteriores

    3. Un Tipo Abstracto de Dato (TAD) permite
        a. que los programan que utilicen la especificación del TAD no tienen que ser cambiados cada vez que la implementación del TAD cambie
        b. obligar a cambiar los programas que utilicen la especificación del TAD cada vez que haya cambios en la implementación del TAD
        c. que los programas implementen directamente las operaciones del TAD para cambiar su comportamiento
        d. ninguna de las anteriores

    4. En verificación de software una falla es
        a. un estado intermedio incorrecto en que entra un programa al ejecutarse
        b. un síntoma de que existe un error
        c. un caso de prueba
        d. ninguna de las anteriores

    5. La verificación formal de programas está basada
        a. en probar los programas con casos de prueba exhaustivos
        b. en utilizar laboratorios de usuarios que prueben recurrentemente los programas
        c. en especificaciones matemáticas (p.e. algebraicas) sobre lo que se espera de las operaciones del programa
        d. ninguna de las anteriores

6. En un TAD pila
    a. es posible insertar elementos por el tope o por el fondo de la pila
    b. solo es posible insertar elementos por el fondo de la pila
    c. es posible acceder de manera directa a cualquier elemento de la pila
    d. ninguna de las anteriores

7. En un TAD cola
    a. es posible eliminar cualquier elemento bajo cualquier circunstancia
    b. solo es posible eliminar el primero de la cola
    c. solo es posible eliminar el último de la cola
    d. ninguna de las anteriores

8. La búsqueda con centinela
    a. tiene el mismo orden de complejidad que la búsqueda sin centinela pero realiza más pasos
    b. realiza una búsqueda binaria y, por tanto, mejora el orden de complejidad con respecto a la búsqueda sin centinela
    c. consume menos memoria que la búsqueda sin centinela
    d. ninguna de las anteriores

9. La búsqueda binaria
    a. tiene orden de complejidad cuadrático
    b. realiza dos pasadas para ordenar los datos
    c. tiene orden de complejidad lineal
    d. ninguna de las anteriores

10. Para un mismo problema disponemos de un algoritmo de orden n^n, otro de orden $n!$, otro de orden 2^n y otro de orden n^{10}. ¿Cuál elegimos?
    a. el polinómico (n^{10})
    b. el de orden n^n
    c. el de orden $n!$
    d. ninguna de las anteriores

11. Dos algoritmos A y B tienen ambos orden superior lineal, el algoritmo A tiene orden inferior 1 y el algoritmo B tiene orden inferior sublineal, ¿Cuál elegimos?
    a. A
    b. B
    c. es irrelevante pues en promedio ambos darán un rendimiento equivalente
    d. ninguna de las anteriores

12. El orden exacto de complejidad (Θ) nos da una idea de
    a. el comportamiento del algoritmo en el mejor de los casos
    b. el comportamiento del algoritmo en el peor de los casos
    c. el comportamiento del algoritmo en el mejor de los casos y en el peor de los casos
    d. ninguna de las anteriores

13. ¿por qué se suele realizar eL análisis de algoritmos centrándonos en el comportamiento para problemas de talla grande?
    a. porque nos permite comprender como se comportan los algoritmos en condiciones extremas
    b. porque en la realidad nunca se presentan problemas de talla pequeña
    c. porque así aseguramos el comportamiento del algoritmo en el mejor de los casos
    d. ninguna de las anteriores

14. El algoritmo quicksort de ordenación
    a. es menos eficiente temporalmente que el algoritmo de la burbuja
    b. es un ejemplo de algoritmo lineal
    c. es un ejemplo de algoritmo cuadrático
    d. ninguna de las anteriores

15. Los algoritmos de fuerza bruta
    a. suelen implicar alto coste computacional
    b. toman una decisión y nunca la reconsideran
    c. realizan una poda del espacio de búsqueda de soluciones
    d. ninguna de las anteriores

16. Un algoritmo voraz
    a. divide el problema en subproblemas de modo recursivo
    b. suele ser más eficiente temporalmente que uno de backtracking
    c. es un caso especial de ramificación y poda
    d. ninguna de las anteriores

17. Un algoritmo voraz
    a. nunca reconsidera una decisión tomada previamente
    b. puede reconsiderar una decisión previa en caso de llegar a un callejón sin salida
    c. es un caso especial de fuerza bruta
    d. ninguna de las anteriores

18. La programación dinámica
    a. consiste en sacrificar tiempo computacional para conseguir ahorrar uso de memoria
    b. consiste en sacrificar memoria para conseguir mejor tiempo computacional
    c. consiste en decidir dinámicamente (en tiempo de ejecución) entre ahorrar memoria o ahorrar tiempo
    d. ninguna de las anteriores

19. La técnica de ramificación y poda...
    a. es una variante de la fuerza bruta
    b. es una variante de los algoritmos voraces
    c. realiza un recorrido informado del espacio de búsqueda
    d. ninguna de las anteriores

20. Los algoritmos de vuelta atrás o backtracking...
    a. son similares en filosofía a los algoritmos fuerza bruta
    b. realizan una búsqueda dentro de un espacio de búsqueda que se suele representar de forma lineal
    c. nunca realizan un recorrido en profundidad del espacio de búsqueda
    d. ninguna de las anteriores

2. (1 pto) En la ETSE, se desea diseñar una aplicación web de recepción de solicitudes para usar un ordenador portátil. Se quiere poder almacenar las solicitudes recibidas por parte de profesores de la manera más efectiva y eficaz. Cuando el ordenador es devuelto por un profesor, se pasa al siguiente que lo hubiese pedido (según el orden establecido por la fecha y hora de solicitud). Responde brevemente a las siguientes preguntas:
    a. ¿Qué TAD usarías para gestionar el conjunto de personas a la espera? ¿por qué?
    b. La gente que puede pedir el ordenador es un conjunto restringido de personas (profesores de la ETSE) que tiene un tamaño reducido y además hay pocos cambios en la plantilla de profesorado. Teniendo en cuenta esto, ¿qué estrategia utilizarías internamente en el TAD para almacenar los datos: memoria contigua o memoria enlazada? ¿por qué?

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

E.T.S.E

PROGRAMACIÓN II

Examen

Profesor: David E. Losada **Mayo 2011**

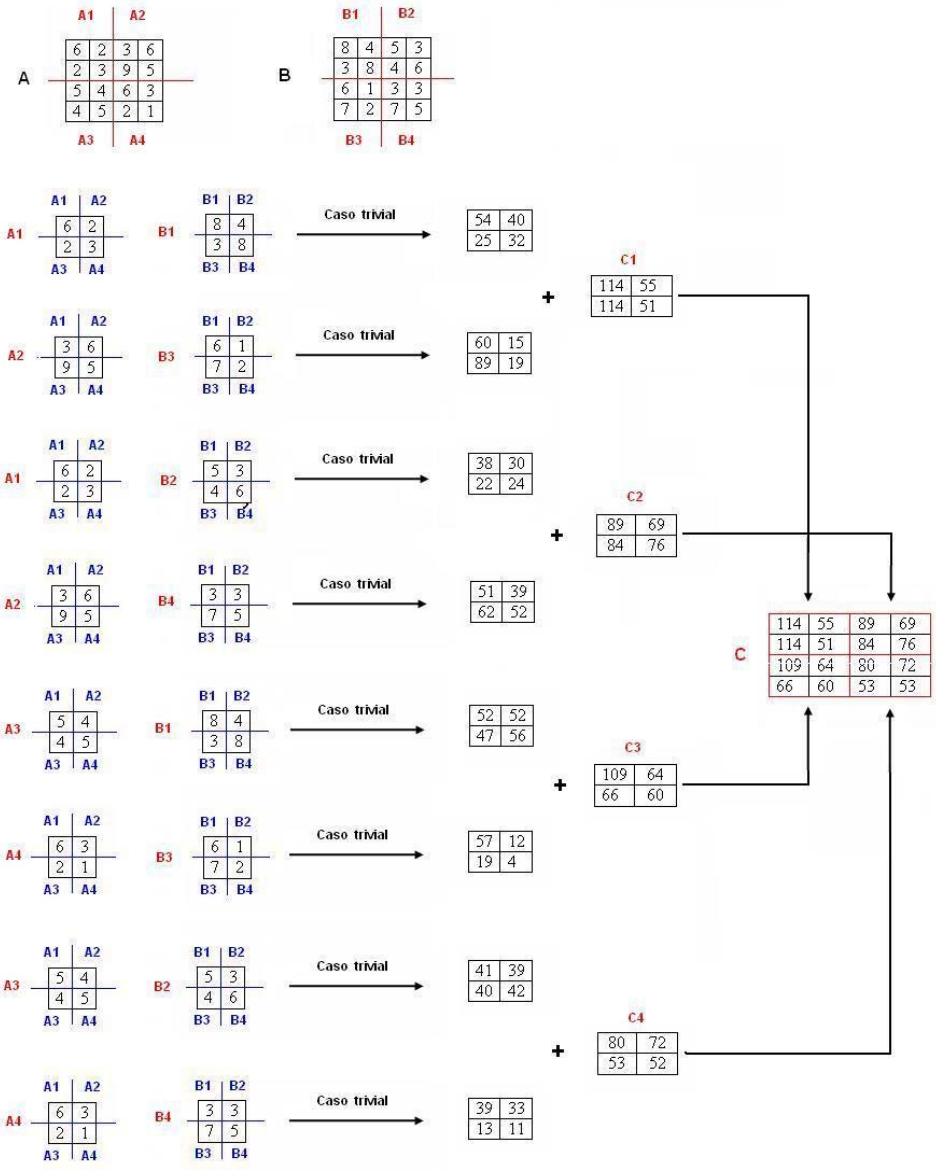
Fecha: 31/5/2011.

Nombre y apellidos:_____

PARTE II (CON MATERIAL)

3. (1 punto) La múltiplicación de matrices cuadradas, en su implementación directa iterativa (que no es divide y vencerás), ¿qué orden de complejidad tiene respecto al tamaño de la matriz en número de filas/columnas? ¿por qué?

   En cambio, la multiplicación divide y vencerás realizada en las prácticas, basada en el esquema de división que figura a continuación, ¿qué orden de complejidad tiene? ¿por qué?



4. (1.5 puntos) Dado los siguientes programas, determina el orden superior (O) e inferior (Ω) de complejidad de los mismos siendo la talla del problema el número N

Programa A:

```
....

int main()
{
....

if (N%55 == 0)
        k = f1(N);
else
        k=f2(N);


}
```

```
float f1(int num)
{
int aux, aux2, aux3;
float rdo=0;

for (aux=0;aux<num;aux++)
  for (aux2=0;aux2<num;aux2++)
    for (aux3=0;aux3<num;aux3++)
       rdo += aux*aux2*aux3;

return rdo;
}
```

```
float f2(int num)
{
int aux,aux2;
float rdo=0;

for (aux=0;aux<num;aux++)
   for (aux2=0;aux2<num;aux2++)
      rdo += aux+aux2;

return rdo;
}
```

Programa B:

```
....

int main()
{
....

k = f1(N);

}
```

```
float f1(int num)
{
float rdo;

if (num == 1)
   return (1.1)

rdo= 2*f1(num-1)*f1(num-2);

return(rdo);
}
```

Programa C:

```
....

int main()
{
....

k = f2(N);

if (k>1000)
   k+=f1(N);

}
```

```
float f1(int num)
{
int aux, aux2;
float rdo=0;

for (aux=0;aux<100;aux++)
  for (aux2=0;aux2<num;aux2++)
       rdo += aux*aux2;

return (rdo);
}
```

```
float f2(int num)
{
int aux;
float rdo=0;

for (aux=0;aux<num;aux++)
  rdo += aux;

return rdo;
}
```

5. (1.5 punto) Construye la especificación formal del TAD Conjunto (que modele conjuntos de elementos todos ellos de tipo *elemento*), con las operaciones crear (crea el conjunto vacío), añadir (añade un elemento al conjunto), borrar (elimina un elemento del conjunto), pertenece (saber si un elemento está en el conjunto) y esvacio (saber si el conjunto está vacío).

6. (2 puntos) Una gestora de reservas de una sala de juntas dispone de tres listas independientes donde se almacena qué personas usaron la sala en el pasado. Una lista almacena los directivos que la usaron, otra lista almacena los administrativos que la usaron y la última lista almacena los informáticos que la usaron. La única información que se guarda de cada persona es su DNI. En ninguna de las listas existen individuos repetidos y tampoco aparece ningún individuo en más de una lista. Para evitar gestionar 3 listas quiere un procedimiento informático que le genere una lista con todos los individuos que han usado la sala en el pasado. Utiliza las operaciones necesarias del TAD Lista para resolverle su problema implementando la función....

```
// rellena los argumentos y tipo que recibe la función
void crea_Lista_Unica(                                    )
{
// rellena las instrucciones necesarias

.....

}
```