NOMBRE:_____APELLIDOS:_____

CUESTIONES (1:00 h) – No se puede utilizar ningún tipo de apuntes o libros

1) Dadas las siguientes funciones iterativas (2 ptos)

```
a)
func sum(v: vector[1..N] de enteros)
       dev v2: vector[1..N] de enteros
{Pre:  N > 0 }
i := 1;
v2[1] := v[1];
{Inv: ???; Dec: ???}
while( i ≠ N ) → v2[i + 1] := v[i + 1] + v[i];
        i := i + 1;
  ]
{Post: ∀α:2 ≤ α ≤ N: v2[α] = v[α] + v[α - 1] ∧
       v2[1] = v[1] }
```

```
b)
func sum_cuad(v: vector[1..N] de reales) dev s: real
{Pre:  cierto }
i := N;
s := 0;
{Inv: ???; Dec: ???}
while( i ≠ 0 )→ s := s + v[i] * v[i];
               i := i - 1;
  ]
{Post: s = ∑α:1 ≤ α ≤ N: v[α] * v[α] }
```

Escribe la función de cota y el invariante:

a) INV:

FC:

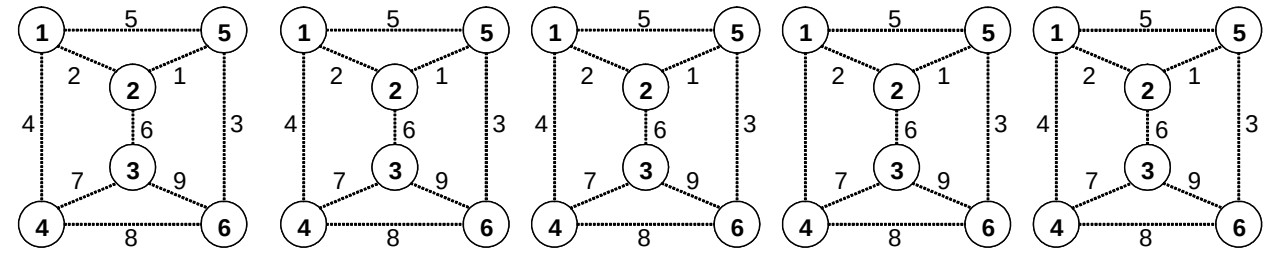
b) INV:

FC:

2) Considera el siguiente algoritmo (es parecido a PRIM) y responde a las siguientes preguntas (3 ptos):

```
ALGORITMO EJERCICIO2
DATOS:     G = (N,A,p)    // Grafo no dirigido con pesos
RESULTADO: T ⊆ A          // Conjunto de arcos de A
AUXILIAR:  F ⊆ A          // Arcos seleccionados en cada iteración
           B ⊆ N          // Nodos utilizados
METODO:
    T ← 0
    B ← 0
    (x,y) ← minimo (A)
    T ← T ∪ {(x,y)}
    B ← B ∪ {x} ; B ← B ∪ {y}
    Repetir
        F ← {(x',y'): x' ∈ N-B ∧ y' ∈ B ∧ p(x',y') > p(x,y)}  //(INST PREGUNTA C)
        (x,y) ← minimo (F)
        T ← T ∪ {(x,y)}
        B ← B ∪ {x}
    Hasta que (B = N) v (F = {})
FEjercicio2
```

a) Dado el siguiente grafo, indica paso a paso como se va construyendo el conjunto de arcos que se generan. (Resalta el arco seleccionado y escribe el valor del conjunto B)(1 pto)



b) ¿Qué 2 condiciones debe de tener el Grafo no dirigido con pesos (G), para que este algoritmo permita obtener el "*árbol de extensión minimal*" del Grafo? (1 pto)

c) ¿Cómo afecta al algoritmo el hecho de cambiar la instrucción marcada por la siguiente? (1 pto)

$F \leftarrow \{(x',y'): x' \in N-B \wedge y' \in B \wedge p(x',y') >= p(x,y)\}$

3) Considera el problema de las Torres de Hanoy para 4 agujas sin restricciones en el movimiento de las agujas y responde (supón que siempre hay un número par de discos):.

a) Escribe un algoritmo que resuelva el problema de forma <u>óptima</u> (1 pto)

b) Si calculamos los costes en función del número de movimientos de los discos. ¿Serán los costes <u>espacial</u> y <u>temporal</u> obtenidos para el caso de 4 agujas la mitad que para el caso de 3 agujas?. Justifica la respuesta (1 pto)

4) Responde verdadero o falso a las siguientes afirmaciones (Las preguntas incorrectas restan ½ punto) (3 ptos)

a. Los algoritmos de vuelta a atrás (backtracking) plantean una forma eficiente de realizar la exploración del espacio de soluciones ..... V o F
b. Se define un montículo de máximos como un árbol binario completo donde cada nodo cumple que no es menor que sus nodos hijos ..... V o F

c. El problema de la "Mochila discreta", donde los objetos no se pueden fraccionar, es un problema típico que se resuelve mediante un algoritmo voraz ..... V o F
d. El coste espacial de los algoritmos de retroceso suele ser muy alto comparado con el resto de tipos de algoritmos recursivos ..... V o F
e. Los algoritmos voraces se basan en la utilización de un criterio local que permita seleccionar un resultado para cada decisión local ..... V o F
f. La poda del árbol de vuelta a atrás consiste en eliminar de forma aleatoria algunas ramas del árbol de exploración de forma que se reduzca el coste temporal ..... V o F
g. El nodo raíz de un montículo de máximos corresponde al elemento mínimo del árbol ..... V o F
h. La vuelta a atrás se plantea como una mejora de la búsqueda exhaustiva y como una alternativa a los algoritmos voraces ..... V o F
i. Un vector ordenado de forma creciente siempre es un montículo de máximos utilizando una representación mediante vectores ..... V o F

PROBLEMAS (1:00) – Se pueden utilizar apuntes y libros

5) Obtén la solución para la siguiente ecuación de recurrencia (5 ptos): $4T(n+1) = T(n-1) + 2^{n+2}$

SOLUCIÓN	T(n) =

Desarrollo:

6) Diseña un algoritmo de ordenación <u>recursivo</u> que se base en la descomposición de un vector de n elementos en dos mitades. Para ordenar la primera mitad vuelve a llamar al algoritmo, sin embargo, la segunda mitad del algoritmo se debe ordenar de forma <u>directa</u>, utilizando el algoritmo de Intercambio (o burbuja) (5 ptos).

a) Escribe el algoritmo recursivo (en seudocódigo) y TODAS las funciones y algoritmos auxiliares que se necesitan (*OJO con el paso de parámetros!!*) (2 ptos)

b) Escribe la ecuación de recursividad del algoritmo si lo que se valora son únicamente comparaciones sobre el vector.(2 pto)

c) Escribe el coste espacial del algoritmo (1 pto)