***Diseño y Paradigmas de Lenguajes - Año 2015***
***Práctico Nro. 3***
***Administración de Memoria***
***Corresponde a Cap. V Abstracción I: Encapsulamiento***
***(Pratt en Español) págs. 203-206 y 216-234.***

**Ejercicio 1.**

Dada la siguiente función en lenguaje C:

```
float cuadratica(int a, int b) {
    const int c = 1;
    static float f;

    float x1, x2, raiz;

    raiz = pow(b,2) - 4 * a * c;

    x1 = (- b + sqrt(raiz)) / 2 * a;
    x2 = (- b - sqrt(raiz)) /2 * a;

    f = pow(a * x1,2) + b * x2 + c;

    return f;
}
```

1. Especifique en qué parte de la plantilla obtenida en la traducción de la función `cuadratica` se almacenarán los objetos de datos: `a, b, c, f, x1` y `x2` cuando se produzca la activación de la función.
2. Suponga que la función es invocada con `cuadratica(m,n)`, considerando que `m y n` son variables enteras con valores 3 y 5 respectivamente. Dibuje la estructura resultante de la activación de la función.
3. Muestre paso a paso cómo van cambiando los objetos de datos del registro de activación desde el comienzo hasta la finalización de la ejecución de la función `cuadratica(3,5)`.

**Ejercicio 2.**

Dado el siguiente código en C:

```
int *p,*q,*s;
p = (int*)(malloc(sizeof(int));
q = (int*)(malloc(sizeof(int));
s = p;
free((void*)p);
*s = 4;
q = p
```

a. Indique si existe algún problema de administración de la memoria. Si su respuesta es afirmativa, rotule la sentencia en la que se provoca el problema, nombre el problema y explique por qué sucede.

b. Considerando la sentencia `*s = 4`, ¿Es posible acceder al objeto de datos apuntado por `s`? ¿Qué consecuencia podría producir esta asignación?

**Ejercicio 3.**

Considere el siguiente fragmento de código en C:

```
int *p,*q,*r,*s;
p = (int *) malloc(sizeof(int));
q = (int *) malloc(sizeof(int));
r = (int *) malloc(sizeof(int));
r = q;
s = q;
free((void *)q);
r = p;
```

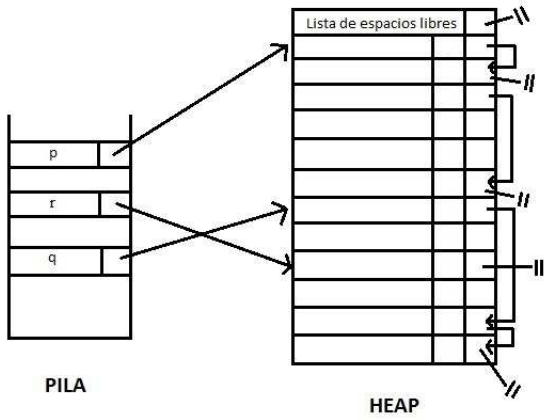
a. Indique si existe algún problema de administración de la memoria, indicando el nombre del problema y explicando porqué se produce.

b. Si se implementa la técnica de recuperación de memoria Contador de Referencias, muestre paso a paso cómo se modifica el heap y la lista de espacios libres en la ejecución de cada sentencia del código.

**Ejercicio 4.**

a. Liste las condiciones que deben cumplirse para poder ejecutar correctamente la etapa de marcado del algoritmo de *recolección de basura*.

b. Suponga que el montículo (heap) se encuentra como muestra la figura y que a continuación se debe aplicar el algoritmo de *recolección de basura*. Muestre gráficamente cómo va cambiando la estructura del heap al aplicar cada paso del algoritmo, en cada una de las estapas.



**Ejercicio 5.**

Sea el siguiente código en C:

```
typedef struct item{
        int elem;
        struct item *h;
        struct item *t;
       } tipoItem;
void main(){
 tipoItem *p,*q;
 p = (tipoItem *)malloc(sizeof(tipoItem));
 p -> h = (tipoItem *)malloc(sizeof(tipoItem));
 q = p -> h;
 q -> h = (tipoItem *)malloc(sizeof(tipoItem));
 p -> t = (tipoItem *)malloc(sizeof(tipoItem));
 q -> t = (tipoItem *)malloc(sizeof(tipoItem));
 p = q;
 p -> t = (tipoItem *)malloc(sizeof(tipoItem)); (*)
}
```

1. Muestre como quedaría el heap y los punteros luego de la ejecución de la última sentencia del código.
2. Suponga que luego de ejecutar la sentencia rotulada con (*) el espacio libre del heap se termina y se invoca automáticamente el algoritmo de *Recolección de Basura*. Muestre claramente paso a paso cómo van cambiando las estructuras del almacenamiento con la ejecución de cada etapa del algoritmo.

**Ejercicio 6.**

Dadas las siguientes declaraciones y sentencias pertenecientes a un trozo de programa en lenguaje C:

```
int* a[4];
int *p;
a[0] = (int*)malloc(sizeof(int)*5);
a[1] = (int*)malloc(sizeof(int)*10);
p = a[1];
a[2] = (int*)malloc(sizeof(int)*8);
a[3] = a[2];
a[1] = (int*)malloc(sizeof(int));
p = a[0];
a[1] = a[2];
free((void*)a[1]);
a[2] = (int*)malloc(sizeof(int));
```

a. Muestre paso a paso cómo cambian las estructuras de almacenamiento luego de la ejecución de cada una de las sentencias. Asuma que se trabaja con un heap de tamaño variable y el tamaño del entero es 4 bytes.

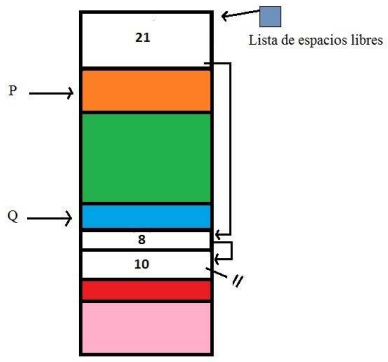
b. Indique si existe algún problema de administración de la memoria.

c. Suponga ahora que la memoria se administra considerando la técnica de recuperación de memoria *Contador de Referencias*. Muestre paso a paso cómo cambian las estructuras de almacenamiento luego de la ejecución de cada una de las sentencias del código.

d. Si se implementara el algoritmo *Recolección de basura*, ¿Qué información se necesita almacenar en cada elemento del heap?

**Ejercicio 7.**

Suponga que se trabaja con un sistema que cuenta con una gestión de almacenamiento en montículo (heap) con elementos de tamaño variable. Considere que en determinado momento de la ejecución de un programa el montículo se encuentra como lo muestra la siguiente figura.




a. Suponga que se realiza un requerimiento de memoria de 9 bytes, determine qué bloque de la lista de espacios libres será asignado y muestre cómo se modifica la lista de espacios libres, según se utilice:
   - Método del mejor ajuste.
   - Método del primer ajuste.

b. Muestre cómo se modifica el montículo si se liberan explícitamente las porciones de memoria referenciadas por `P` y `Q` y luego se realiza la compactación considerando el enfoque:
   - Compactación parcial.
   - Compactación total (cabal).

c. Explique cuál de las implementaciones del tipo puntero (dirección absoluta y relativa) es más conveniente para realizar el punto `b`.