Diseño y Paradigmas de Lenguajes - Año 2014

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)*10);
q = (int*)(malloc(sizeof(int)*3);
p = q;
s = p;
free((void*)q);
*s = 5;
```

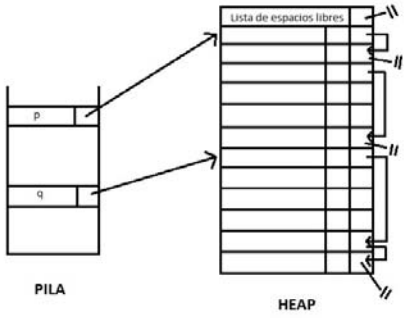
a. Explique qué problema de administración de memoria se produce luego de la ejecución de `p = q`. Justifique. Si existiera forma de recuperar el espacio afectado al problema, ¿Sería posible determinar su ubicación?

b. Explique qué problema de administración de memoria se produce luego de la ejecución de `free((void*)q)`. Justifique su respuesta.

c. Considerando la sentencia `*s = 5`, ¿Es posible acceder al objeto de datos apuntado por `s`? ¿Qué consecuencia podría producir esta asignación?

Ejercicio 3.

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.



Ejercicio 4.

Considerando cada uno de los 4 mecanismos de gestión de almacenamiento (estático, basado en pilas y basado en montículo (heap) con elementos de tamaño fijo y variable), dibuje cómo se encuentran las estructuras de almacenamiento en la fase *asignación inicial*.

Ejercicio 5.

Considere el siguiente fragmento de código en C:

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

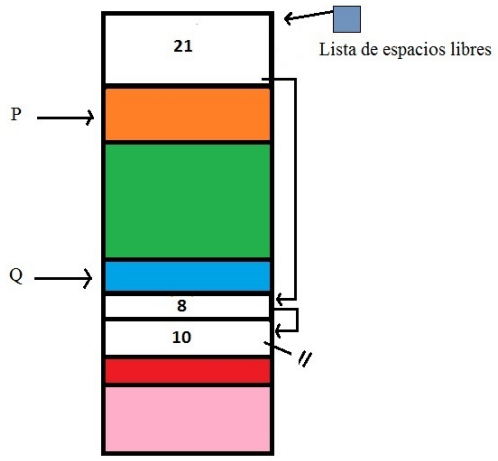
a. Indique luego de la ejecución de qué sentencias se genera basura y/o referencias desactivadas.

b. Considere que se implementa la técnica de recuperación de memoria Contador de Referencias, muestre cómo se modifica el montículo y la lista de espacios libres en la ejecución de cada sentencia del código.

c. Suponga que luego de la ejecución de la última sentencia se agota el espacio en la lista de espacios libres y se ejecuta el algoritmo de *Recolección de basura*. Muestre paso a paso cada una de las etapas involucradas en el algoritmo y como quedan el montículo y la lista de espacios libres.

Ejercicio 6.

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 7 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).