

*Fundamentos del Diseño de Lenguajes de Programación - 2024*  
*Práctico Nro. 4*  
*Administración de Memoria*  
Corresponde a Cap. VI págs. 238-245 y Cap. X págs. 415-432 del Pratt

### Ejercicio 1.

Dada la siguiente función en lenguaje C:

```
#include<math.h>
#include <stdlib.h>
int F1(int v) {
    static int resu = 5;
    const int e = 10;
    resu += abs(v - e);
    return resu;
}
```

- a. Especifique en qué parte de la plantilla obtenida en la traducción de la función F1 se almacenarán los objetos de datos: constante literal 10, v, resu y e cuando se produzca la activación de la función.
- b. Suponga que la función es invocada con F1(m), considerando que m es una variable entera con valor 7. Dibuje la estructura resultante de la activación de la función.

### Ejercicio 2.

Considere el siguiente fragmento de código en C:

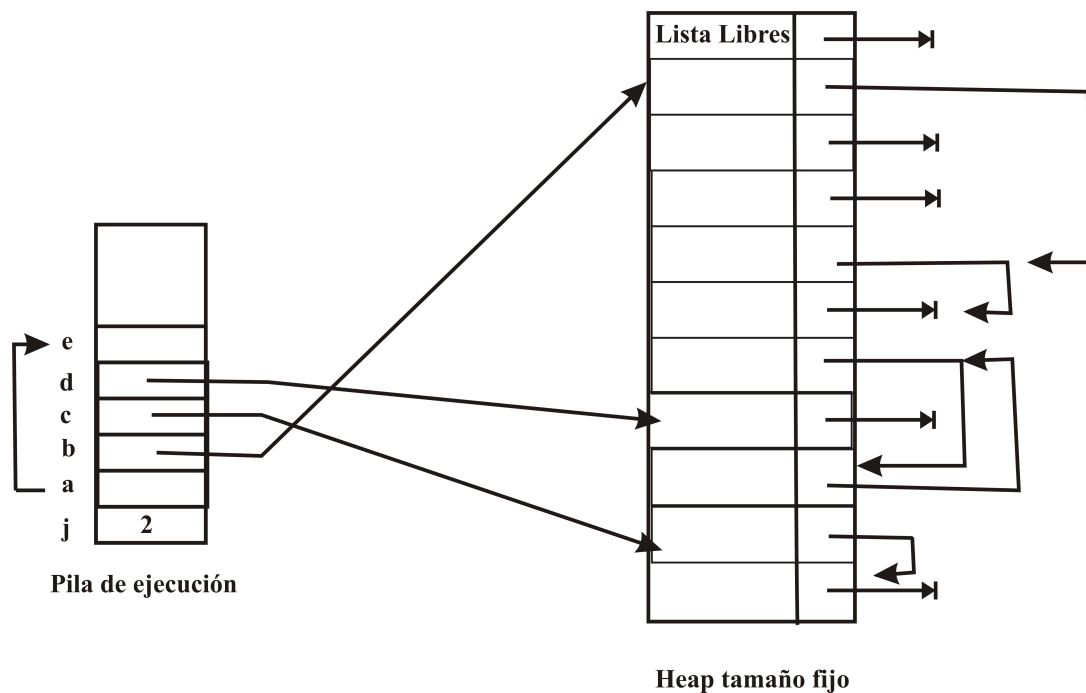
```
#include<stdlib.h>
int main() {
    int *p, *q, *r, *t, *u;
    int a = 6;
    p = (int*)malloc(sizeof(int));
    q = (int*)malloc(sizeof(int));
    t = (int*)malloc(sizeof(int));
    p = q;
    u = t;
    r = (int*)malloc(sizeof(int));
    r = &a;
    free((void *)t);
    return 0;
}
```

- a. Indique si existe algún problema de administración de la memoria.
- b. Ejecute paso a paso el código con la herramienta online:  
<https://pythontutor.com/c.html#mode=edit> seleccione en C/C++ details: show memory addresses y compruebe la respuesta del ítem anterior.
- c. Considerando que luego del free se adiciona la sentencia \*u = 4;, ¿Es posible acceder al objeto de datos apuntado por u? ¿Qué consecuencia podría producir esta asignación?

- d. Suponga que en el Heap (montículo) se utiliza el mecanismo de recuperación llamado 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, hasta la última inclusive. Considere que el tamaño del entero es 2 bytes.

### Ejercicio 3.

Suponga que la pila de ejecución y un heap de tamaño fijo se encuentra como se muestra en la siguiente figura:



- Liste las condiciones que deben cumplirse para poder ejecutar correctamente la etapa de marcado del algoritmo de *Recolección de Basura*.
- Ante la imposibilidad de satisfacer un requerimiento de memoria (lista de espacios libre vacía), se ejecuta el algoritmo de *Recolección de Basura*, muestre gráficamente cómo va cambiando la estructura del heap y la lista de espacios libres al aplicar cada etapa del algoritmo. Haga diferentes gráficos para cada etapa explicada.

#### Ejercicio 4.

A partir del siguiente programa en lenguaje C y considerando un heap con bloques de tamaño fijo:

```
#include<stdlib.h>
typedef struct{
    int    cod;
    float* dato[6];
}Tinfo;
void main(){
    Tinfo Elem;
    int k;
    for(k = 0; k < 6; k += 2)
        Elem.dato[k] = (float *)malloc(sizeof(float));
    Elem.dato[1] = Elem.dato[2];
    Elem.dato[4] = Elem.dato[1];
    Elem.dato[5] = (float *)malloc(sizeof(float));
    free((void*) Elem.dato[2]);
    Elem.dato[0] = Elem.dato[5];
}
```

- Muestre como quedaría el heap de tamaño fijo, la pila de ejecución y los punteros luego de la ejecución de la última sentencia del código. Suponga que el tamaño del tipo entero es 2 bytes, el flotante 5 y el puntero a flotante de 6 bytes. La estructura Elem comienza en la dirección  $\alpha = 1500$ .
- Indique si existe algún problema de administración de la memoria. Especifique la sentencia donde se produce y justifique.
- Suponga que luego de ejecutar la última sentencia, el espacio libre del heap de tamaño fijo se termina y se ejecuta el algoritmo de *Recolección de basura*. Muestre paso a paso cómo cambian las estructuras de almacenamiento para mostrar, cada una de las etapas involucradas en la ejecución del algoritmo.

#### Ejercicio 5.

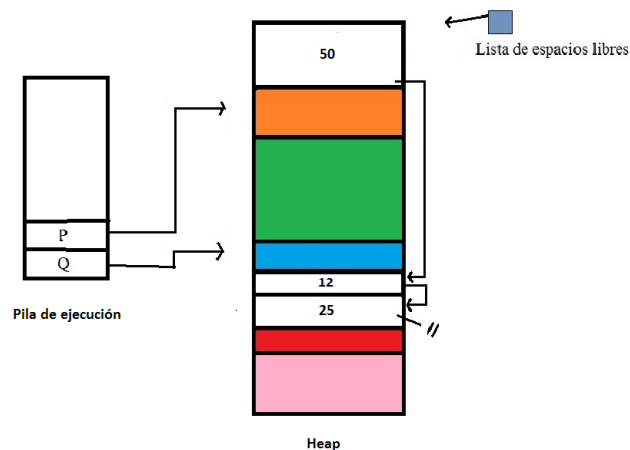
A partir del siguiente programa en lenguaje C y considerando un heap con bloques de tamaño variable:

```
#include<stdlib.h>
typedef struct{
    float    pre;
    char *    sig;
}Tdato;
int main(){
    Tdato t[5];
    t[0].sig = (char *)malloc(sizeof(char)* 2);
    t[1].sig = (char *)malloc(sizeof(char)* 5);
    t[2].sig = t[0].sig;
    free((void*) t[0].sig); // *
    t[3].sig = (char *)malloc(sizeof(char)* 4);
    t[4].sig = (char *)malloc(sizeof(char)* 7);
    t[3].sig = t[4].sig; // #
    return 0;
}
```

- Muestre como quedaría el heap de tamaño variable, la pila de ejecución y los punteros luego de la ejecución paso a paso de todas las sentencias del código. Suponga que el tamaño del tipo char es 1 byte, el flotante es 4 y el puntero a char de 5 bytes.
- Ejecute paso a paso el código con la herramienta `pythontutor` y compruebe la respuesta del ítem anterior.
- Suponga 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 hasta la sentencia marcada con (\*) inclusive.
- Suponga que luego de ejecutar la sentencia marcada con (#), el espacio libre del heap se termina y se ejecuta el algoritmo de *Recolección de basura*. Muestre paso a paso cómo cambian las estructuras de almacenamiento para mostrar, cada una de las etapas involucradas en la ejecución del algoritmo. ¿Qué información se necesita almacenar en cada elemento del heap para poder aplicar el algoritmo?

### Ejercicio 6.

Suponga que se trabaja con un sistema que cuenta con una gestión de almacenamiento en 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.



- Suponga que se realiza un requerimiento de memoria de 20 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.
- 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).