PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

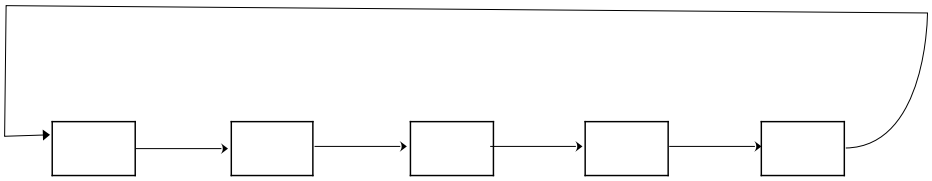
INF239 SISTEMAS OPERATIVOS
Semestre 2023-1
Laboratorio 4

1) (4 puntos – nombre del archivo a entregar: *xfree.ods*) El segmento de código, mostrado a continuación, ha sido tomado de la función `xfree()` :

```
for (p= freep; !(bp > p && bp < p->s.ptr); p = p->s.ptr)
          if (p >= p->s.ptr && (bp > p || bp < p->s.ptr))
                    break;
```

Este lazo busca el lugar donde se encuentra el bloque que se desea liberar.

Dada una lista como la que se muestra a continuación:



Emplee el archivo *xfree.ods* para dibujar los posibles casos donde se puede encontrar el bloque. El bloque dibujado rellénelo de color amarillo. Adicionalmente debe indicar la condición que se debe cumplir, por cada caso, para justificar el lugar donde ha ubicado el bloque. Si la condición no se especifica no se asignan puntos.

2.1) (7 puntos – nombre de los archivos a entregar: *Makefile, xalloc.c, xrun.c, xalloc.h*)

Se ha modificado el programa *xrun.c* de la siguiente forma:

```c
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <math.h>
#include "xalloc.h"

int main(void)
{ unsigned int x,unidad,base;
  void *pt;


  unidad=1024;
  x=0;

  printf("El tamaño del header es:%d bytes\n",sizeh());
  do {
    base=pow(2,x)+.5;
    if((pt=xmalloc(base*unidad))) {
       fprintf(stdout,"Se solicita %d bytes Se proporciona %d headers ubicados en %p\n",base*unidad,size(pt),pt);
       xprintq();
    }
    else
       fprintf(stderr,"No hay suficiente memoria\n");
    x++; }
  while(x<=6);
  exit(0);
}
```

donde las funciones `sizeh()`, `size(pt)` y `xprintq()` se deben implementar en *xalloc.c*

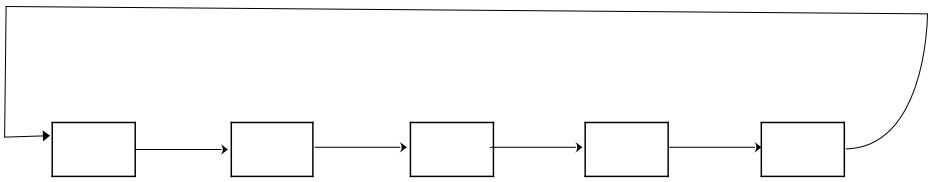
```
sizeh()       # (1 punto) devuelve el tamaño del Header en bytes
size(pt)      # (3 puntos) devuelve cuántos Headers le ha asignado xmalloc
xprintq()     # (3 puntos) imprime la lista de bloques libres
```

La salida debe ser análoga a la siguiente:

```
El tamaño del header es:16 bytes
Se solicita 1024 bytes Se proporciona 65 headers ubicados en 0x55d43f53ac00
0x55d43ea98090 0
0x55d43f537000 959
Se solicita 2048 bytes Se proporciona 129 headers ubicados en 0x55d43f53a3f0
0x55d43ea98090 0
0x55d43f537000 830
Se solicita 4096 bytes Se proporciona 257 headers ubicados en 0x55d43f5393e0
0x55d43ea98090 0
0x55d43f537000 573
Se solicita 8192 bytes Se proporciona 513 headers ubicados en 0x55d43f5373d0
0x55d43ea98090 0
0x55d43f537000 60
Se solicita 16384 bytes Se proporciona 1025 headers ubicados en 0x55d43f53b010
0x55d43f537000 60
0x55d43ea98090 0
Se solicita 32768 bytes Se proporciona 2049 headers ubicados en 0x55d43f53f020
0x55d43f537000 60
0x55d43ea98090 0
Se solicita 65536 bytes Se proporciona 4097 headers ubicados en 0x55d43f547030
0x55d43f537000 60
0x55d43ea98090 0
```

2.2) (7 puntos – nombre de los archivos a entregar: *xrun.ods*)
Haciendo uso de la salida de su programa anterior, haga un esquema de la lista de bloques libres y los bloques usados. Emplee las celdas, del archivo *xrun.ods* para representar los bloques. Deben de elaborar un esquema por cada solicitud, 7 en total. Los bloques ocupados rellénelos de color amarillo. Dentro de las celdas escriba el tamaño de los bloques en *headers*. **En el esquema se debe notar si los bloques están juntos o separados. Esto influirá en la calificación.**

3) (2 puntos – nombre del archivo a entregar: *xrun.c*)
Modifique el programa *xrun.c* para que al final la lista tenga la siguiente representación:



Los bloques libres deben ser al menos 3. Puede usar `xmalloc()` y `xfree()`. Las solicitudes no tienen que estar necesariamente en un bucle.

Porf. Alejandro T. Bello Ruiz