TD – MALLOC – Partie 2 – B – Liste chainée

1 – Commentez et expliquez le code ci-dessous

#include <unistd.h>

#include <stdlib.h>

// ?? – 1

#define ENTETE\_SIZE (ALIGN(sizeof(bloc\_entete)))

// ?? – 2

typedef struct bloc\_entete{

size\_t taille;

unsigned short libre :1;

} bloc\_entete;

// ?? – 3

bloc\_entete\* ADR\_PREMIER\_BLOC=0;

// ?? – 4

typedef struct bloc\_entete

{ size\_t taille;

struct bloc\_entete\* suivant\_ptr ;

struct bloc\_entete\* precedent\_ptr ;

} bloc\_entete ;

// ? – 5

bloc\_entete\* ADR\_PREMIER\_BLOC\_LIBRE=0;

void\* myalloc(size\_t t){

// ?? – 6

size\_t blk\_size = ALIGN(t+ENTETE\_SIZE);

// ?? – 7

if(ADR\_PREMIER\_BLOC\_LIBRE == 0){

void\* bloc\_total=(void\*)sbrk(blk\_size);

void\* bloc\_utilisateur=(void\*)bloc\_total+ENTETE\_SIZE

bloc\_entete\* entete = (bloc\_entete\*)bloc\_total;

entete->taille=blk\_size;

entete->suivant\_ptr=0;

entete->precedent\_ptr=NULL;

ADR\_PREMIER\_BLOC = (bloc\_entete\*)entete;

}

// ?? - 8

else{ if((bloc\_entete\*)ADR\_PREMIER\_BLOC\_LIBRE->suivant\_ptr==0){

void\* bloc\_total=(void\*)sbrk(blk\_size);

void\* bloc\_utilisateur=(void\*)bloc\_total+ENTETE\_SIZE

bloc\_entete\* entete = (bloc\_entete\*)bloc\_total;

entete->taille=blk\_size;

entete->suivant\_ptr=0;

entete->precedent\_ptr=0;

ADR\_PREMIER\_BLOC = (bloc\_entete\*)entete;

}

// ?? - 9

else{ bloc\_entete\* current = (bloc\_entete\*)ADR\_PREMIER\_BLOC\_LIBRE->suivant\_ptr;

int alloc=0;

while( current!=0 && alloc==0){

if(current->taille >= blk\_size){

if(current->suivant!=0){

current->suivant\_ptr->precedent\_ptr = (bloc\_entete\*)current->precedent\_ptr;

}

if(current->precedent\_ptr!=0 || current->precedent\_ptr!=NULL){

current->precedent\_ptr->suivant\_ptr = (bloc\_entete\*)current->suivant\_ptr

}

alloc=1;

void\* bloc\_utilisateur=(void\*)current+ENTETE\_SIZE;

}

else{ current = (bloc\_entete\*)current->suivant\_ptr;

}

}

// ?? - 10

if(alloc==0){

void\* bloc\_total=(void\*)sbrk(blk\_size);

void\* bloc\_utilisateur=(void\*)bloc\_total+ENTETE\_SIZE

bloc\_entete\* entete = (bloc\_entete\*) bloc\_total;

entete->taille=blk\_size;

entete->suivant\_ptr=0;

entete->precedent\_ptr= (bloc\_entete\*)current->precedent\_ptr;

}

}

}

return bloc\_utilisateur;

}

void myfree3(void\* ptr){

// ?? - 11

bloc\_entete\* entete\_courante =(bloc\_entete\*)ptr-ENTETE\_SIZE;

if(entete\_courante->precedent\_ptr!=0 || entete\_courante->precedent\_ptr!=NULL){

entete\_courante->precedent\_ptr->suivant\_ptr=(bloc\_entete\*)entete\_courante;

}

// ?? - 12

if(entete\_courante->suivant\_ptr!=0){

entete\_courante->suivant\_ptr->precedent\_ptr=(bloc\_entete\*)entete\_courante;

}

}

void main(int argc, char\* argv[]){

// Partie a compléter

}

|  |  |
| --- | --- |
| Commentaires |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |

2 – Complétez le main

void main(int argc, char\* argv[]){

// partie à compléter

}