4 - Les segments de la mémoire

1. Placez-vous dans un répertoire **`M2101/MemorySegments`**, récupérez sur le site Web du cours le fichier **`memory_segments.c`** (dont le code est fourni en annexe), et complétez le tableau suivant à l'aide de **`gdb`** :

arrêt	*variable*	*adresse*	*valeur*	*segment*
l. 11	**`main`**			
	`to_minutes`			
	`counter`			
	`c`			
l. 18	**`total`**			
l. 11	**`counter`**			
	`c`			
l. 25	**`total`**			
	`a`			
	`ptr_a`			
	`*ptr_a`			
	`LENGTH`			
	`ptr_LENGTH`			
	`width`			
	`tab`			
	`*tab`			
	`tab[0]`			
	`tab[1]`			
	`*(tab + 1)`			
l. 26	**`tab`**			
	`*tab`			
l. 28	**`tab`**			
	`*tab`			

2. Faites un dessin où vous placerez toutes les variables du tableau.
3. Expliquez la différence entre **`counter`** et **`c`**.
4. Que vaut **`(tab+1)-tab`** ?
5. Expliquez la différence entre **`free(tab)`** et **`tab=NULL`**.
6. Pourquoi est-il conseillé de faire **`tab=NULL`** après **`free(tab)`** ?
7. Qu'arrive-t-il si on fait **`tab=NULL`** sans avoir fait **`free(tab)`** ?

Listing 1 – memory_segments.c

```c
#include <stdlib.h>

const int LENGTH = 10;
int width = 5;

int to_minutes (int hours, int minutes) {
        static int counter = 0;
        ++counter;
        int c = 0;
        ++c;
        return 60 * hours + minutes;
}

int main(void) {
        int a = 18;
        int b = 33;
        int total = to_minutes (a, b);
        total = to_minutes (b, a);

        int *ptr_a = &a;
        const int *ptr_LENGTH = &LENGTH;
        int *tab = malloc (LENGTH * sizeof(int));
        tab[0] = 31;
        *(tab + 1) = 43;
        free (tab);
        tab = NULL;

        return 0;
}
```