·Scope of variables + where are they stored ?
· new vs malloc

CSC209H Worksheet: malloc Basics

1. Each time a variable is declared or memory is otherwise allocated, it is important to understand how much memory is allocated, where it will be allocated and when it will be de-allocated. Complete the table below. (Note: some of the programs allocate more than one block of memory.)

Code Fragment	Space?	Where?	De-allocated when?

`int main() {`
`→ int i;`
`}`
	sizeof(int)	stack frame for `main`	when program ends

`int fun() {`
`    float i;`
`}`
`int main() {`
`    fun();`
`}`
	size of (float)	stack frame for fun	when fun returns

`int fun(char i) {`
`    ...`
`}`
`int main() {`
`    fun('a');`
`}`
	→ 1	"	"

`int main() {`
`    char i[10] = {'h','i'};`
`}`
	10 bytes	main stack	when prog. ends

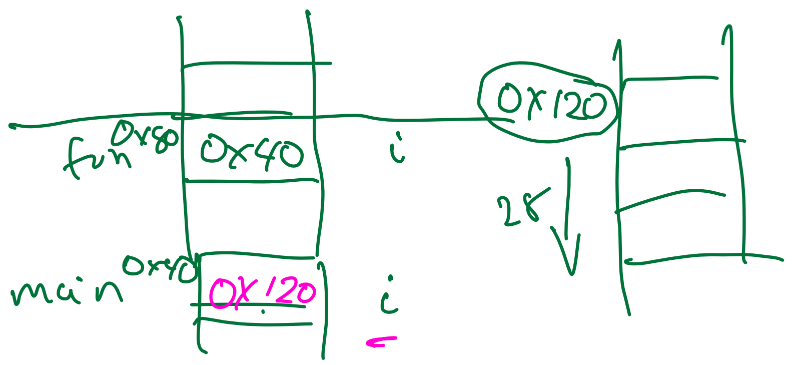
`int main() {`
`    char *i;`
`}`
	8 bytes	main stack	"

`int main() {`
`    int *i;`
`}`
	8 bytes	"	"

`int fun(int *i) {`
`    ...`
`}`
`int main() {`
`    int i[5] = {4,5,2,5,1};`
`    fun(i);`
`}`
	→ 8
	→ 5×4
		fun
		main
			when fun returns
			when prog ends

`int main() {`
`  int *i;`
`  i = malloc(sizeof(int));`
`}`
	→ 8
	→ 4
		main
		heap
			when prog ends
			→ when we call free

`void fun(int **i) {`
`  *i = malloc(sizeof(int)*7);`
`}`
`int main() {`
`    int *i;`
`    fun(&i);`
`    free(i);`
`}`
	→ 8
	→ 28
	→ 8
		fun
		heap
		main
			→ when fun returns
			→ free
			→ when main ends



CSC209H Worksheet: malloc Basics

2. Trace the memory usage for the program below up to the point when `initialize` is about to return. We have set up both stack frames for you, and the location of the heap.

```
#include <stdio.h>
#include <stdlib.h>


// Initialize two parallel lists.
void initialize(int *a1, int *a2, int n) {
    for (int i = 0; i < n; i++) {
        a1[i] = i;
        a2[i] = i;
    }
}

int main() {
    int numbers1[3];
    int *numbers2 = malloc(sizeof(int) * 3);

    initialize(numbers1, numbers2, 3);

    for (int i = 0; i < 3; i++) {
        printf("%d %d\n",
               numbers1[i], numbers2[i]);
    }

    free(numbers2);
    return 0;
}
```

note
no label!

Section	Address	Value	Label
Heap	0x23c	2a0	
	0x240		
	0x244		
	0x248		
	⋮	⋮	
stack frame for initialize n	0x454	7dc	
a2	0x458	7e0	
	0x45c	7e4	
a1	0x460	7e8	
	0x464	7ec	
	0x46c	7f0 ⋮	
i	0x470	7fc	
stack frame for main i	0x474	81c	
numbers 2	0x478	820	2a0
	0x47c	824	
	0x480	828	
numbers1	0x484	82c	
	0x488	830	
	0x48c	834	

gdb
addresses