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();`
`}`
	sizeof(float)	stack frame for fun	when fun returns

`int fun(char i) {`
`    ...`
`}`
`int main() {`
`    fun('a');`
`}`
	sizeof(char)	stack frame for fun	when fun returns

`int main() {`
`    char i[10] = {'h','i'};`
`}`
	10* sizeof(char)	stack frame for main	when program ends

`int main() {`
`    char *i;`
`}`
	sizeof(char *)	"	"

`int main() {`
`    int *i;`
`}`
	sizeof(int *)	"	"

`int fun(int *i) {`
`    ...`
`}`
`int main() {`
`    int i[5] = {4,5,2,5,1};`
`    fun(i);`
`}`
	→ sizeof(int *) →
	→ 5* sizeof(int) →
		stack frame for fun →
		stack frame for main →
			when fun returns
			when program ends

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

`void fun(int **i) {`
`  *i = malloc(sizeof(int)*7);`
`}`
`int main() {`
`    int *i;`
`    fun(&i);`
`    free(i);`
`}`
	→ sizeof(int **) →
	→ 7* sizeof(int) →
	→ sizeof(int *) →
		stack frame for fun →
		heap →
		stack frame for main →
			when fun returns
			free(i)
			when program 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.

~~reserved by malloc~~

```
#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;
}
```

Section	Address	Value	Label
Heap	0x23c	0	
	0x240	1	
	0x244	2	
	0x248		
	⋮	⋮	
stack frame for initialize	0x454	0x474	a1
	0x458		
	0x45c	0x23c	a2
	0x460		
	0x464	3	n
	0x46c	~~0~~ ~~1~~ ~~2~~ 3	i
	0x470		
stack frame for main	num1 0x474	0	numbers1[0]
	0x478	1	
	0x47c	2	
	0x480	0x23c	numbers2
	0x484		
	0x488	~~0~~ ~~1~~ ~~2~~ 3	i
	0x48c		