CSC209H Worksheet: Stacks and Heaps

1. Trace the memory usage for the program below. We have set up both stack frames for you, and the location of the heap.

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

int *mkarray1(int a, int b, int c) {
    int arr[3];
    arr[0] = a;
    arr[1] = b;
    arr[2] = c;

    int *p = arr;
    return p;
}

// Code for other_function() omitted.

int main() {

    int *ptr = mkarray1(10, 20, 30);
    other_function();
    printf("%d %d %d\n", ptr[0], ptr[1], ptr[2]);
}
```

int *arr = malloc(3 * sizeof int)

return arr

free(ptr);

Section	Address	Value	Label
Heap	0x23c	10	
	0x240	20	
	0x244	30	
	0x248		
	⋮	⋮	
stack frame for mkarray1	0x454	10	a
	0x458	20	b
	0x45c	30	c
arr: 0x23c	0x460	10	arr
	0x464	20	
	0x46c	30	
	0x470	0x460	p
	0x474		
	0x478		
	0x47c		
stack frame for main	0x480	0x460	ptr ← 0x23c
	0x484		
	0x488		
	0x48c		

The problem is that when mkarray returns, the stack frame is no longer valid, and when other_function is called, its local variable may overwrite that part of the stack.

2. The program in part 1 will not work correctly. Notice the call to `other_function`. Explain to your partner why the program doesn't work. Fix the `mkarray1` function, and trace it again. Fixes in green

3. Once you've fixed the code, add a statement to your program to deallocate the memory on the heap as soon as possible.

CSC209H Worksheet: Stacks and Heaps

4. Trace the memory usage for the program below. We have set up the stack frame for you, and the location of the heap.

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

/* Build an array in dynamic memory to hold
   multiples of x from x to x*x.
   Return a pointer to this array.
*/
int *multiples(int x) {
    int *a = malloc(sizeof(int) * x);
    for (int i=0; i < x; i++) {
        a[i] = (i+1) * x;
    }
    return a;
}

int main() {
    int *ptr;
    int size = 3;

    ptr = multiples(size);

    for (int i=0; i<size; i++) {
        printf("%d\t", ptr[i]);
    }
    printf("\n");

    return 0;
}
```

Section	Address	Value	Label
Heap	0x224	3	
	0x228	6	
	0x22c	9	
	0x230		
	0x234		
	0x238		
	0x23c		
	0x240		
	0x244		
	⋮	⋮	
stack frame for multiples	0x470	3	x
	0x470	0x224	a
	0x474		
	0x478	~~0~~ ~~x~~ ~~3~~	i
stack frame for main	0x47c	0x224	ptr
	0x480		
	0x484	3	size
	0x488	~~0~~ ~~x~~ ~~3~~	i
	0x48c		

5. Change the main function so that it calls `multiples` and prints the array in a loop with sizes of 3, 4, and 5. Besides the changes described, do not make any other changes or additions to the code.

6. Trace the memory usage of your changed program. Explain the problem to your partner and then fix it by adding calls to deallocate the memory.

Memory leak

Memory Leak

Memory that cannot be accessed because the address is not "saved" in any varible

Dangling Pointer - AKA "Use after free"

making use of a pointer to memory that has been freed

free(ptr)

*ptr = ?