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;
}


int main() {

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

Section	Address	Value	Label
Heap	0x23c		
	0x240		
	0x244		
	0x248		
	⋮	⋮	
stack frame for mkarray1	0x454		
	0x458		
	0x45c		
	0x460		
	0x464		
	0x46c		
	0x470		
	0x474		
	0x478		
	0x47c		
stack frame for main	0x480		
	0x484		
	0x488		
	0x48c		

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.

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		
	0x228		
	0x22c		
	0x230		
	0x234		
	0x238		
	0x23c		
	0x240		
	0x244		
	⋮	⋮	
stack frame for multiples	0x470		
	0x470		
	0x474		
	0x478		
stack frame for main	0x47c		
	0x480		
	0x484		
	0x488		
	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.