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

```c
#include <stdio.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", arr[0], arr[1], arr[2]);
    return 0;
}
```

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

```c
#include <stdio.h>

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

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

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 mkarray function, and trace it again. **Answer shown above**

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

```c
free (ptr);
```
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\n", ptr[i]);
    }
    printf("\n");
    return 0;
}
```

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

```c
#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;
    for (int j = size; j <= 5; j++) {  
        ptr = multiples(size);
        for (int i = 0; i < size; i++) {
            printf("%d\n", ptr[i]);
        }
        printf("\n");
    }
    return 0;
}
```

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. `free(ptr);`