Exercise 1

- Use **size** shell command to determine the size of text, data and bss segments of any of your programs. Save the output to file ex1.txt

Exercise 2

- Write a C program that dynamically allocates memory for an array of N integers, fills the array with incremental values starting from 0, prints the array and deallocates the memory. Program should prompt the user to enter N before allocating the memory.

Exercise 3

- Complete the following code template according to the comments. The purpose of the program is to create an initial array of a user-specified size, then dynamically resize the array to a new user-specified size.

Exercise 4

- Write your own realloc() function using malloc() and free()
  - realloc() changes the size of the memory block pointed to by ptr to size bytes. The contents will be unchanged in the range from the start of the region up to the minimum of the old and new sizes.
  - Newly allocated memory will be uninitialized
  - If ptr is NULL, the call is equivalent to malloc(size)
  - If size is equal to zero, the call is equivalent to free(ptr)
  - Unless ptr is NULL, it must have been returned by an earlier call to malloc(), calloc() or realloc()

Exercise 5

- Find and fix all the code that generates segmentation faults

```
#include <stdio.h>
int main() {
    char **s;
    char foo[] = "Hello World";
    *s = foo;
    printf("s is %s\n",s);
    s[0] = foo;
    printf("s[0] is %s\n",s[0]);
    return(0);
}
```