Module 6 solutions

1. See quiz answers here.
2. Do exercise 5 (Ch. 13, p. 309) in the textbook:

```c
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

void capitalize_a(char *s) {
    if (!s) return;
    size_t n = strlen(s);
    for (size_t k=0;k<n;k++) s[k] = toupper(s[k]);
}

void capitalize_b(char *s) {
    if (!s) return;
    for (;*s != '\0'; s++ ) *s = toupper(*s);
}
```

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void capitalize_a(char *s);
void capitalize_b(char *s);

int main(int argc, char const *argv[])
{
    if(argc != 2) {
        printf("Usage: %s \"string\"\n",argv[0]);
        return EXIT_SUCCESS;
    }

    // Find length and allocate space for a copy of argv[1]
    size_t n = strlen(argv[1]);
    char * str = malloc((n+1)*sizeof(*str));
    if (!str) return EXIT_FAILURE;

    // Copy argv[1] and apply capitalize_a()
    strncpy(str,argv[1],n);
    capitalize_a(str);
    printf("Part (a): %s\n", str);

    // Copy argv[1] and apply capitalize_b()
```

```
    strncpy(str,argv[1],n);
    capitalize_b(str);
    printf("Part (b): %s\n", str);

    free(str);
    return EXIT_SUCCESS;
}
```

3. Do exercise 12 (Ch. 22, p. 583) in the textbook:

   The problem is that `fgetc` is called *twice* in the loop, so the program only counts periods at odd indices. This can be fixed by using a single `fgetc()` call in the loop:

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

int count_periods(const char *filename)
{
    FILE *fp;
    int n = 0;
    int c;
    if ((fp = fopen(filename, "r")) != NULL)
    {
        //while (fgetc(fp) != EOF)
        //    if (fgetc(fp) == '.') n++;
        while ((c = fgetc(fp)) != EOF)
        {
            if (c == '.')
                n++;
        }
        fclose(fp);
    }
    return n;
}
```

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

int count_periods(const char *filename);

int main(int argc, char const *argv[])
{
    FILE *fp;
    if (argc != 2)
    {
        printf("Usage: %s filename\n", argv[0]);
```

```
        return EXIT_SUCCESS;
    }
    if ((fp = fopen(argv[1], "r")) == NULL) // Check if file exists
    {
        fprintf(stderr, "Error reading file %s\n", argv[1]);
        return EXIT_FAILURE;
    }
    fclose(fp);
    printf("Found %d periods in file %s\n",
            count_periods(argv[1]), argv[1]);
    return 0;
}
```

4. Autolab solutions will be available on DTU Learn after the deadline.
5. To print the capacity, we can insert the line `printf("Capacity: %zu\n", a->capacity);` before printing the elements of the array.
6. Create a program that uses the MSP Tools library to read a two-dimensional array from a text file.

```
#include <stdlib.h>
#include <stdio.h>
#include "array2d.h"

int main(int argc, char const *argv[])
{
    if(argc != 2) {
        printf("Usage: %s filename\n",argv[0]);
        return EXIT_SUCCESS;
    }
    array2d_t *A = array2d_from_file(argv[1]);
    if (!A) {
        fprintf(stderr,"Error reading file %s\n",argv[1]);
        return EXIT_FAILURE;
    }
    array2d_print(A);
    array2d_dealloc(A);
    return EXIT_SUCCESS;
}
```

7. Create a program that uses the MSP Tools library to read a sparse matrix from a text file.

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

int main(int argc, char const *argv[])
{
```

```
    if(argc != 2) {
        printf("Usage: %s filename\n",argv[0]);
        return EXIT_SUCCESS;
    }
    coo_t *A = coo_from_file(argv[1]);
    if (!A) {
        fprintf(stderr,"Error reading file %s\n",argv[1]);
        return EXIT_FAILURE;
    }
    coo_print(A);
    coo_dealloc(A);
    return EXIT_SUCCESS;
}
```