# 02635 Fall 2016 — Module 8

## Homework

- Read chapter 6 pp. 219-255 and chapter 12 pp. 489-519 in "Beginning C"

## Exercises

1. Take this quiz to test your understanding of strings.

2. Write a program that evaluates a function (say, $f(x) = e^{-x} + 0.5x$) at $n \geq 2$ points between a lower limit $x_l$ and and upper limit $x_u$ $(x_l < x_u)$, i.e., the program should evaluate the function at

$$x_i = x_l + (i-1)\frac{x_u - x_l}{n-1}, \quad i = 1, \ldots, n.$$

   The program should create a new text file (say, `data.txt`) with the $n$ value pairs $(x_i, f(x_i))$ on separate lines and with $x_i$ and $f(x_i)$ separated by a space. The number of points $n$ and the lower and upper limits $x_l$ and $x_u$ should be user inputs.

   **Remark**: You can load and plot the data in MATLAB using the following commands:

```
>> load data.txt
>> plot(data(:,1),data(:,2))
>> xlabel('x')
>> ylabel('f(x)')
```

   You can also plot the data using Gnuplot. Here is a basic example:

```
set terminal postscript
set output "plot.eps"
plot "data.txt" using 1:2 with lines notitle
set xlabel "x"
set ylabel "f(x)"
```

3. A sparse matrix is a matrix in which many entries are zero. Storing such a matrix as a full two-dimensional array is often inefficient in terms of both *space* (i.e., memory) and *time* (i.e., computational cost). For

example, a diagonal matrix of order $n$ has zeros in all off-diagonal positions, and hence it can be stored as a one-dimensional array of length $n$ corresponding to the $n$ diagonal elements. Furthermore, multiplication with a diagonal matrix is much cheaper computationally than multiplication with a general dense matrix. More generally, a sparse matrix $C$ of size $m \times n$ can be represented as a set of triplets of the form $(i, j, C_{ij})$. For example, the matrix

$$C = \begin{bmatrix} 1.0 & 0 & 0 & 4.0 \\ 0 & 2.0 & 3.0 & 0 \\ 5.0 & 0 & 6.0 & 7.0 \end{bmatrix}$$

has seven nonzeros and can be represented in triplet form as

$$\{(1, 1, 1.0), (3, 1, 5.0), (2, 2, 2.0), (2, 3, 3.0), (3, 3, 6.0), (1, 4, 4.0), (3, 4, 7.0)\}.$$

When working with sparse matrices in triplet form, it is convenient to define a structure that stores both the dimensions of the matrix as well as three arrays corresponding to the row indices, the column indices, and the nonzero values. Such a structure may look like this:

```
/* Structure representing a sparse matrix in triplet form */
struct sparse_triplet {
    unsigned long m;   /* number of rows     */
    unsigned long n;   /* number of columns  */
    unsigned long nnz; /* number of nonzeros */
    unsigned long * I; /* pointer to array with row indices    */
    unsigned long * J; /* pointer to array with column indices */
    double * V;        /* pointer to array with values         */
};
```

- Write a function that can read a sparse matrix in triplet form from a text file in which each line corresponds to a nonzero element. Your function should allocate and return a `sparse_triplet` stucture. You may assume that the first line of the text file contains the dimensions $m$ and $n$ as well as the number of nonzeros, i.e., the text file that corresponds to the above example should look like this:

```
3 4 7
1 1 1.0
3 1 5.0
2 2 2.0
2 3 3.0
3 3 6.0
1 4 4.0
3 4 7.0
```

The function should convert the indices to 0-based indices so that row indices are between $0$ and $m-1$ (instead of $1$ to $m$) and column indices are between $0$ and $n-1$ (instead of $1$ and $n$).

**Hint**: Your function prototype could look like this:

```
struct sparse_triplet * read_sparse(const char * filename);
```

Start by allocating memory for a `sparse_triplet` structure, and read the first line of the file to determine `m`, `n`, and `nnz`. Then allocate memory for the row/columns indices (`I` and `J`) and the nonzero values (`V`), i.e., two `unsigned long` arrays of length `nnz` and a `double` array of length `nnz`. Finally, read the remaining `nnz` lines of the file and fill the arrays. Do not forget to close files that you open.

- Write a short program to test your function. Remember to deallocate the memory that has been allocated by your function.

4. Write a function that takes a `sparse_triplet` structure and a file name as input and creates a text file with the matrix in triplet form. The text file should follow the format described in the previous exercise (i.e., the first line should contain the dimensions `m n nnz` and the remaining `nnz` lines should contain the triplets). Write a short program to test your function.

   **Hint**: Your function prototype could look like this:

   ```
   int write_sparse(const char * filename, struct sparse_triplet * A);
   ```

# Optional exercises

1. Write a function that performs the matrix-vector multiplication

$$y := Ax + y$$

where $A$ is a sparse matrix of size $m \times n$ in triplet form and $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are vectors (i.e., arrays). Use the following prototype for your function:

```c
/* Sparse matrix-vector product */
void spmv(const struct sparse_triplet * A, const double * x, double * y);
```

Write a short program to test your function.

2. Write a program that (i) reads a list of names from a file, (ii) prompts the user to enter a grade for each user, and (iii) writes the names and grades to a new file.