Programming in C I, Group B
Jacobs University Bremen
Dr. Kinga Lipskoch
Instructor: Xu He

Course: JTSK-320111
September 28th, 2017

Assignment 5 - More with Loops and Strings, Dynamic Memory Allocation, Multidimensional Arrays

- The problems of this assignment must be solved in C.
- The TAs are grading solutions to the problems according to the following criteria: http://minds.jacobs-university.de/sites/default/files/uploads/teaching/CProgrammingFall17/Grading-Criteria-c.pdf

Problem 5.1 *A table* (1 point)

Presence assignment, due by 18:30 h today

Write a program that prints on the screen a table (you can assume that a column with not be wider than 12 characters, the values should be aligned to the right, floating point numbers should be printed with a precision of 3 places) where each line consists of an integer value, its square, and its cube separated by space. Ask the user to input from the keyboard two integers as the lower and upper limits for the table (i.e., at which value to start and where to stop with both as including values).
Use a `for` loop for solving this problem.
You can safely assume that the entered integer values for the upper and lower limits will be valid.

Problem 5.2 *Word as diagonal* (1 point)

Presence assignment, due by 18:30 h today

Write a program where you read a string from the standard input. You can safely assume that the string will be not longer than 80 characters. Make sure that you are allowed to enter a string containing spaces as well. Print the string on the screen in the following manner:

```
H
 e
  l
   l
    o
```

Problem 5.3 *Using switch and calling functions* (2 points)

Write a program where you can read from the keyboard up to 10 integers into an array. A negative value ends the input loop and the negative value is not part of the array. You can assume that no more than 10 integers will be read.
Then by using a `switch` statement, pressing `m` (and 'Enter') computes the harmonic mean of the array (and prints the result), `h` prints the highest number in the array, `l` prints the minimal number in the array, `s` prints the sum of all elements in the array and `n` prints the number of elements in the array. Write functions for each task of the `switch`. The return value of these functions must be printed from the `main()` function.
The prototype of the function computing the harmonic mean should have the following form:

```
double harmonic_mean(int arr[], int num);
```

The formula for computing the harmonic mean of an array $(x_i)_{i=1,\ldots,n}$ with n elements is:

$$hmean = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

You can safely assume that the input will be valid.

Problem 5.4 *Determine consonants in string I* (1 point)

Write a function `int count_consonants(char str[])` that determines and returns the number of consonants in a given string.
Then write a simple main function where you can repeatedly enter a string and then the number of consonants is determined and printed on the screen (from the `main()` function). If the entered string is empty (it will contain only a `'\n'` then) the program should stop its execution. You should be able to enter strings containing spaces.
You can assume that the entered string will be not longer than 100 *characters.*

Bonus Problem 5.5 *Determine consonants in string II* (1 point)

Modify the function `int count_consonants(char str[])` from your solution for **Problem 5.4** to walk through the string using a pointer and address arithmetic. You can reuse the `main` function from the previous problem's solution.

Problem 5.6 *Dynamic memory allocation I* (1 point)

Write a function `double maxdiff(double arr[], int n)` that determines and returns the difference between the largest and smallest elements (in this order) of an array of doubles.
Then write a program where you first read a number n and then n doubles that are stored in an array a. Call the function from above and print the result on the screen. Use dynamic memory allocation and deallocation.
You can safely assume that the input will be valid.

Bonus Problem 5.7 *Dynamic memory allocation I* (2 points)

Write a program which allocates memory for an array of floats entered from the keyboard having n elements (value read also from the keyboard). Write and call a function which determines and prints on the screen the two smallest values within this array. At the end of the program make sure that the memory will be released.
Solve the problem without sorting the array or without iterating through the array more than one time.
You can safely assume that the input will be valid.

Problem 5.8 *Main diagonal of matrix* (1 point)

Write a program which declares a two dimensional array of integers having at most 50 rows and 50 columns. Read from the keyboard an integer n representing the number of rows and the number of columns of the matrix (i.e., square matrix). Then read the values of the matrix row by row and column by column. Then write and call a function for printing the values of the matrix in its form. Also write and call a function which prints on the screen the elements of the matrix which are on the main diagonal.
You can safely assume that the input will be valid.

Testcase 5.8: input

```
3
1
2
3
4
5
6
7
8
9
```

Testcase 5.8: output

```
The entered matrix is:
1 2 3
4 5 6
7 8 9
The main diagonal is:
1 5 9
```

Bonus Problem 5.9 *Secondary diagonal of matrix* (1 point)

Modify your solution for **Problem 5.8** such that the elements on the secondary diagonal (i.e., the other diagonal) are printed on the screen.

Problem 5.10 *Passing by address* (1 point)

Write a function `void sumdiffproddiv(double a, double b, double *sum, double *diff, double *prod, double *div)` that computes and "returns" by address the sum, the difference (first minus second), the product and the division (first divided by second) of the two doubles.
Also write a simple test program to check that your function works correctly.
You can safely assume that the input will be valid. You can assume that `b` *will be non-zero.*

Problem 5.11 *Walk the string* (1 point)

Write a function `int count_char(char *str, char c)` that counts the occurrences of the character `c` in the string `str`.
Write a program that tests this function. You should be able to process a string of arbitrary length. First you read `n`, the maximal number of characters the string will have, then you dynamically allocate memory for a string of this size, then the string is read from the keyboard.
Determine the occurrence of the letters `'B'`, `'p'`, `'i'`, `'D'`, and `'z'`. For each character the output should be as follows:

```
The character 'B' occurs 3 times.
```

You can safely assume that the input will be valid.

Problem 5.12 *String functions I* (1 point)

Write a function `void myreplace(char *str, char c, char e)` that replaces the character `c` by the character `e`.
Write a program to test the function. The program should repeatedly read a string (up to 70 chars), a character to be replaced and then the replacing character until you enter "quit" for the string. Your program should print the strings on the screen before and after the replacement.
You can safely assume that the input will be valid.

Bonus Problem 5.13 *String functions II* (1 point)

Write a function `void mixcase(char *str)` that replaces lowercase characters by uppercase and uppercase characters by lowercase. All other characters are not changed.
Use the functions `isupper()`, `islower()`, and potentially `toupper()` and `tolower()` which are declared in `ctype.h`. Use the man pages or online documentation to see how these functions work. Write a program to test the function. The program should repeatedly read a string (up to 80 chars) until you enter "exit" for the string. For every string the lowercase to uppercase and uppercase to lowercase conversion should be printed on the screen.
You can safely assume that the input will be valid.

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` without any warnings (You can use `gcc -Wall -o program program.c`). Insert suitable comments (not on every line . . .) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader.
  Each program **must** include a comment on the top like the following:

```
/*
   JTSK-320111
   a5_p1.c
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://grader.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to `x.he@jacobs-university.de` **with a subject line that begins with JTSK-320111**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Thursday, October 5th, 10:00 h.