Programming in C and C++
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH-230-A
September 24th, 2019

Assignment 4 - More with Loops and Strings, Dynamic Memory Allocation, Multidimensional Arrays

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 4.1 *A table for circles* (1 point)

Presence assignment, due by 11:00 AM today **Graded automatically with testcases only**
Language: C

Write a program that prints a table where each line consists of a value `x`, the area of the circle with radius `x`, and the perimeter of the circle with radius `x` (i.e., three values separated by space). Ask the user to input from the keyboard the lower and upper limits as well as a step size for the table (i.e., at which value to start and where to stop, and the increment from one line to the next).
You can safely assume that the upper and lower limits, and the step size will be valid.
Use a `for` loop for solving this problem.
Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 4.1: input

```
1.5
3
0.5
```

Testcase 4.1: output

```
1.500000 7.068583 9.424778
2.000000 12.566371 12.566371
2.500000 19.634954 15.707963
3.000000 28.274334 18.849556
```

Problem 4.2 *Word as zig zag* (1 point)

Presence assignment, due by 11:00 AM today **Graded automatically with testcases only**
Language: C

Write a program where you read a string (which may contain spaces) from the standard input. You can safely assume that the string will not be longer than 50 characters. Print the string on the screen as in the following testcase.
Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 4.2: input

```
Hello world
```

Testcase 4.2: output

```
H
 e
l
 l
o

w
 o
r
 l
d
```

Problem 4.3 *Using switch and calling functions* (2 points)

Due by Monday, September 30th, 23:00 **Graded manually**

Language: C

Write a program where you can read up to 15 floats from the keyboard 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 15 integers will be read.

Then by using a `switch` statement, pressing `m` (and '`Enter`') computes the geometric mean of the array (and prints the result), `h` determines and prints the highest number in the array, `l` determines and prints the smallest number in the array and `s` determines and prints the sum of all elements in the array. Write functions for each task. The result of these functions must be printed from the `main()` function.

The prototype to compute the geometric mean should have the following form:

```
float geometric_mean(float arr[], int num);
```

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

$$\mathtt{gmean} = \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$$

You can safely assume that the input will be valid.

Problem 4.4 *Determine consonants in string I* (1 point)

Due by Monday, September 30th, 23:00 **Graded automatically with testcases only**

Language: C

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 can safely assume that the entered string will be not longer than 100 characters and will be valid.

Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 4.4: input

```
Hello world
this
last
```

Testcase 4.4: output

```
Number of consonants=7
Number of consonants=3
Number of consonants=3
```

Bonus Problem 4.5 *Determine consonants in string II* (1 point)

Due by Monday, September 30th, 23:00 **Graded manually**

Language: C

Rewrite the function `int count_consonants(char str[])` from your solution of the previous problem to walk through the string using a pointer and address arithmetic. Reuse your `main()` function from the previous problem's solution.

Problem 4.6 *Dynamic memory allocation* (2 points)

Due by Monday, September 30th, 23:00 **Graded manually**

Language: C

Write a program which allocates memory for an array of integers 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 greatest values within this array. At the end of the program make sure that the memory will be released.

You can safely assume that the input will be valid. Solve the problem without sorting the array or without iterating through the array more than one time.

Problem 4.7 *Under the main diagonal of matrix* (1 point)

Due by Monday, September 30th, 23:00 **Graded automatically with testcases only**

Language: C

Write a program which declares a two dimensional array of integers having at most 30 rows and 30 columns. Read from the keyboard an integer `n` representing the number of rows and the number of columns of the 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 under the main diagonal.

You can safely assume that the input will be valid.

Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 4.7: input

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

Testcase 4.7: output

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

Bonus Problem 4.8 *Under the secondary diagonal of matrix* (1 point)

Due by Monday, September 30th, 23:00 **Graded automatically with testcases only**

Language: C

Modify your solution for the previous problem such that the elements under the secondary diagonal (i.e., the other diagonal) are printed on the screen.

You can safely assume that the input will be valid.

Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 4.8: input

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

Testcase 4.8: output

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

Problem 4.9 *Dynamic Array* (1 point)

Due by Monday, September 30th, 23:00 **Graded manually**

Language: C

Write a function `int prodminmax(int arr[], int n)` that determines and returns the product of the smallest and largest elements of an array of integers.

Then write a program where you first read an integer `n` and then `n` integers that are stored in an array called `arr`. Test the function above and print on the screen the results from the `main()` function. Use dynamic memory allocation and deallocation.

You can safely assume that the input will be valid.

Problem 4.10 *Passing by reference* (1 point)

Due by Monday, September 30th, 23:00 **Graded manually**

Language: C

Write a function `void proddivpowinv(float a, float b, float *prod, float *div, float *pwr, float *invb)` that computes and "returns" by reference the product, the division of the two floats as well as the result of a^b and $1/b$.

Also write a simple test program to check that your function works correctly (by printing on the screen the results from the `main()` function).

You can safely assume that the input will be valid and that b *will not be* 0.

Problem 4.11 *Walk the string* (1 point)

Due by Monday, September 30th, 23:00 **Graded manually**

Language: C

Write a function `int count_insensitive(char *str, char c)` that counts the occurrences of the character `c` in the string `str` in the case insensitive manner (i.e., no difference between uppercase and lowercase letters). Write a program that tests this function. You should be able to process a string of arbitrary length. First you can dynamically allocate a string of maximal length of 50 characters, then you read the string from the keyboard, then you allocate memory for another string of the correct size, copy the original string into the new string and then deallocate the memory occupied by the first string. You may use the functions `tolower()` or `toupper()` which are in `ctype.h`. Use the man page to see how these functions work.

You can safely assume that the input will be valid.

Determine the occurrence of the characters `'b'`, `'H'`, `'8'`, `'u'`, and `'$'`. For each character the output should be as follows, for example:

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

Problem 4.12 *String functions I* (1 point)

Due by Monday, September 30th, 23:00 **Graded manually**

Language: C

Write a function `void replaceAll(char *str, char c, char e)` that replaces all occurrences of the character `c` by the character `e`.

Write a program to test the function. The program should repeatedly read a string (up to 80 chars), a character to be replaced and then the replacing character until you enter "`stop`" for the string. Your program should repeatedly print the character to be replaced, the replacing character and the strings on the screen from the main function before and after the replacement.

You can safely assume that the input will be valid.

Bonus Problem 4.13 *String functions II* (1 point)

Due by Monday, September 30th, 23:00 **Graded automatically with testcases only**

Language: C

Write two functions: `void uppcase(char *str)` that replaces lowercase characters by uppercase characters and all other characters are not changed, and `void lowcase(char *str)` that replaces uppercase characters by lowercase characters and all other characters are not changed.

Use the functions `isupper()` and `islower()` which are in `ctype.h`. Use the man page to see how these functions work.

Write a program to test the two functions. The program should repeatedly read a string (up to 90 chars) until you enter "`exit`" for the string. For every string the conversions after calling each of the two functions should be printed on the screen.

You can safely assume that the input will be valid.

Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 4.13: input

```
One?
twO!
This is a sentence.
exit
```

Testcase 4.13: output

```
uppcase=ONE?
lowcase=one?
uppcase=TWO!
lowcase=two!
uppcase=THIS IS A SENTENCE.
lowcase=this is a sentence.
```

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` or `g++` depending on the problem without any errors or warnings (You can use `gcc -Wall -o program program.c` or `g++ -Wall -o program program.cpp`). Insert suitable comments (not on every line ...) to explain what your program does.
- 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:

```
/*
   CH-230-A
   a4_p1.[c or cpp or h]
   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
  `k.lipskoch@jacobs-university.de` **with a subject line that begins with CH-230-A**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- 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 Monday, September 30th, 23:00.