Assignment 13 - Streams, Multiple Inheritance and Exceptions

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
  https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 13.1 *Copy a file* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++

Write a program which reads from the standard input the name of a file (e.g., `"input.txt"`). Your program should use C++ streams to create a copy of this file with the name as the original file name concatenated with `"_copy"` and then the original extension (e.g., `"input_copy.txt"`).
You can assume that the input will be valid and the necessary input file has a valid content if existing.

Problem 13.2 *`Complex` class with streams* (2 points)

Due by Monday, December 2nd, 23:00 **Graded manually**
Language: C++

Adapt and extend your previous source code for working with complex numbers (`Complex.h`, `Complex.cpp` and `testComplex.cpp`) such that the operators + for adding, − for subtracting, * for multiplying two complex numbers, = for assigning, << and >> for input and output streams are overloaded. Your testing program (`testComplex.cpp`) should read two complex numbers from the files called `in1.txt` and `in2.txt`, then compute their sum, the difference and the product, and print the results on the screen as well as into a file called `output.txt`. You have the freedom to set the structure of the input files.
You can assume that the input of the testing program will be valid and the necessary input files have a valid content if existing.

Problem 13.3 *Concatenate `n` files into new file* (2 points)

Due by Monday, December 2nd, 23:00 **Graded manually**
Language: C++

Write a program which reads from the standard input an integer value `n`, followed by `n` file names. Your program has to concatenate the content of the `n` files and write the result into the file called "`concatn.txt`" using binary read and write. Add a `'\n'` to separate the contents of the different files inside the resulting file. The operation of the concatenation is successful only if all files are existing and their opening was successful.
You can assume that the input will be valid and the necessary input files have a valid content if existing.

Problem 13.4 *Multiple inheritance I* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**
Language: C++

Consider the following source file:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/minheritance1.cpp`
Compile and run the file. If you find any compilation errors, explain their reason as comments in the code. Then fix the errors such that the program compiles and runs. Explain the motivation of your modifications in the code and their effects on the execution.

Problem 13.5 *Multiple inheritance II* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Consider the following source file:

`https://grader.eecs.jacobs-university.de/courses/320142/cpp/minheritance2.cpp`

Compile and run the file. If you find any compilation errors, explain their reason as comments in the code. Then fix the errors such that the program compiles and runs. Explain the motivation of your modifications in the code and their effects on the execution.

Bonus Problem 13.6 *`Matrix` and `Vector` classes* (3 points)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Extend your previous source code for working with vectors (`Vector.h` and `Vector.cpp`) and write a `Matrix` class (`Matrix.h`, `Matrix.cpp`) for operations with matrices. If you did not write a `Vector` class for one of the previous bonus problems, then write one with minimal functionality as needed for this problem. Then write a testing program which illustrates operations between matrices and vectors (`testmatvec.cpp`) using the operations described as in the following.

- Overload the operators << and >> to be able to enter matrices and vectors from the standard input and from files (e.g., `in1.txt`, `in2.txt`, etc.), and to send matrices and vectors to the standard output and to files (e.g., `out1.txt`, `out2.txt`, etc.). You have the freedom to set the structure of the input files.
- Overload the operator * for computing the product of a vector and a matrix, and a matrix and a vector. For simplicity reasons you can consider a vector to be either a row vector or a column vector such that mathematical multiplication with a matrix is possible. By this convention, the result of the multiplication is also a vector (row vector or column vector).

Your implementation has to check if the operations between matrix and vector, and vector and matrix are valid or not (concerning the compatibility between a vector and matrix concerning the amount of elements in them). Besides this you can assume that the input of the testing program will be valid, the necessary input files have a valid content if existing.

Problem 13.7 *Out of range exception* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Write a program which creates a vector of integers and adds the value 8, 20 times into the vector. Include the library `<vector>` and then create a `vector` object which stores the 20 values from above. Then write a `try` and `catch` block in which your code should try to access the 21^{th} element from the vector using the `at()` method. The exception you should catch should be of type `out_of_range`. In the `catch` block use `cerr` to print to the standard error stream the type of the exception by calling the redefined `what()` method inherited from the `exception` class.

Bonus Problem 13.8 *Simple different exceptions* (2 points)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Write a program and a function with an integer parameter which can throw four exception types: a `char`, an `int`, a `bool`, and your own exception class called `OwnException` derived from `exception`. If the parameter of the function is 1 then the character `'a'` should be thrown, if it's 2 then the integer `12` should be thrown, if it's 3 then the `bool` value `true` should be thrown, and in the default case an `OwnException` with the string `"Default case exception"` should be thrown. You should overwrite the `what()` method for the `OwnException` class by returning the string `"OwnException\n"`. Call the function in the `main` function in its four versions and catch the corresponding exceptions. In the `catch` blocks you should print to the standard error stream `cerr` the string `"Caught in main: "` followed by the value of the thrown exception. In the case of `OwnException` print the string returned by the `what()` method.

Bonus Problem 13.9 *Car exceptions* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Write a `Garage` class that has a `Car` (i.e., object of a second class) that is having troubles with its `Motor` (i.e., object of a third class). Use a function-level try block in the `Garage` class constructor to catch an exception (thrown from the `Motor` class with the string "`This motor has problems`") when its `Car` object is initialized. Throw a different exception with the string "`The car in this garage has problems with the motor`" from the body of the `Garage` constructor's handler and catch it in the `main` function.

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.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a13_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.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.
- 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 Monday, December 2nd, 23:00.

Assignment 12 - Inheritence, `static` Members, Operator Overloading, Polymorphism

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules: https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 12.1 *`Hexagon` class* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++
Download the files:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.h`
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.cpp`
Write a class called `Hexagon` which is derived from the `RegularPolygon` class. A hexagon has a side and a color. Provide methods for computing the perimeter and the area of a hexagon. For each property appropriate setter and getter methods need to be provided (i.e., it should not be possible to manipulate data directly). The class should also a parametric constructor, a copy constructor and a destructor.

Note, that the area of a hexagon of side `t` can be computed by using the formula $\frac{3\sqrt{3}}{2}t^2$.
Name the files `Shapes.h` and `Shapes.cpp`. Then write a testing program `testHexagon.cpp` that:

a) creates a blue hexagon that has the side of 9,

b) creates a green hexagon that has the side of 15,

c) creates a copy for the second hexagon using the copy constructor, and

d) computes the perimeter and area of all three hexagons and prints the results on the screen.

Problem 12.2 *`TournamentMember` class* (2 points)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++
Imagine that you are in the design stage of a software system for handling the data of the participants of a major soccer tournament. As different roles will be present (players, coaches, referees, etc.) you are required to develop a class handling the data of a generic league member. For each person the following data is at least needed

- first name as character array (36 characters characters including `'\0'`)
- last name as character array (36 characters)
- date of birth as character array (11 characters, storage format is `yyyy-mm-dd`)

In addition the whole team is located somewhere, so `location` is an additional `static` property of the class.
Design and implement a class for holding these data. In addition add at least two other general properties to this class.
The class, which will be called `TournamentMember`, should provide constructors (empty and parametric), a destructor and also a copy constructor. The class should also provide `inline` setter and getter methods (either inside or outside of the class).

Moreover, in order to carry out the functionality of the application, the following methods are required:

- a method which prints the information of a tournament member on the screen,
- a method which changes the `location`.

Also all constructors and the destructor should print a short informational message on the screen, such that you can see which is being called when.
You should provide three files: a header file named `TournamentMember.h` with the declaration of the class, a file named `TournamentMember.cpp` with its definition, and an additional file called `testTournamentMember.cpp` with a `main()` function which tests the functionality of the class.
The needed data can be initialized in the code from the `main()` function.

Problem 12.3 *`Player` class* (1 point)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++

A `Player` class should be derived from the `TournamentMember` class. It holds additional properties such as number, position, number of goals scored, and whether the player is left-footed or right-footed. For each property appropriate setter (except the number of goals scored) and getter methods need to be provided as inline methods, and it should not be possible to manipulate data directly. An appropriate constructor to set all properties on creation should be provided as well as a copy constructor that creates a correct copy of a player, and a destructor. Also all constructors and the destructor should print a short informational message on the screen such that you can see which is being called when.
Also the following methods are required:

- a method which prints all the information of a player on the screen,
- a method which increments the number of goals scored by a player.

Add code to the files `TournamentMember.h`, `TournamentMember.cpp`, and write a testing program named `testPlayer.cpp` that tests the functionality of the `Player` class. Create three players with different properties. Then move all players to the location `"Hamburg"`.
The needed data can be initialized in the code from the `main()` function.

Bonus Problem 12.4 *`Referee` class* (3 points)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++

Write a class named `Referee` derived from the `TournamentMember` class. This class should have the following additional properties:

- `int yellowCardCount;`
- `Player *yellowCardList[40];`
- `int redCardCount;`
- `Player *redCardList[40];`

And the following methods should be implemented:

- `bool addToYellowCardList(Player *p);`
- `bool addToRedCardList(Player *p);`

If the player `p` is not yet on the yellow card list then it should be added to it, but if the player `p` is already on the yellow card list then the player should be removed from the yellow card list and should be added to the red card list. If the player `p` is not yet on the red card list then it should be added to it, but if the player `p` is already on the red card list then nothing should happen. Both methods should return `true` if the adding was successful and `false` if not.
Add code to the files `TournamentMember.h`, `TournamentMember.cpp`, and write a testing program named `testReferee.cpp` that tests the functionality of the `Referee` class. Create a referee and some players. Your code should illustrate the referee actions for adding players to the list of players with yellow cards and to the list of players with red cards.
You can assume that the input or the setting of the data will be valid.

Bonus Problem 12.5 *Choosing a random color* (1 point)

Due by Monday, November 25th, 23:00 **Graded manually**

Language: C++

Write a function which randomly chooses one color from `RED`, `BLACK`, `VIOLET` and `BLUE`. Write a program which calls this function 15 times and prints the chosen color on the screen. You can adapt and modify the program below.

The following code snippet simulates a die throw 10 times. Use an array of strings for the colors and then randomly choose a color by its index.

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main()
{
    int die;
    int count = 0;
    int randomNumber;
    // init random number generator
    srand(static_cast<unsigned int>(time(0)));
    while (count < 10) {
        randomNumber = rand();
        die = (randomNumber % 6) + 1;
        cout << count << ": " << die << endl;
        count++;
    }
    return 0;
}
```

Problem 12.6 *Fractions I* (2 points)

Due by Monday, November 25th, 23:00 **Graded manually**

Language: C++

- As a starting point, use the files `fraction.h, fraction.cpp, testfraction.cpp` (which you can download from:
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/fraction.h`,
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/fraction.cpp`,
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/testfraction.cpp`).
- Replace the method `print()` by an overloaded operator << such that you can use `cout <<` for printing a fraction on the screen.
- Overload the operator >> such that you can enter from the keyboard a fraction using `cin >>`. Check the validity of the input (you can assume that the numerator and denominator will be numbers).
- Overload the operators * and / for computing the multiplication and division of two fractions.
- In your testing program you should then be able to enter two fractional numbers (using `cin >>`), then the product and quotient are printed on the screen (one per line using the overloaded operator and `cout <<`).

Use the suggestions from slide 10 (Lecture 12) for choosing to write member methods or friend functions.

You can assume that the input will be valid.

Problem 12.7 *Fractions II* (2 points)

Due by Monday, November 25th, 23:00 **Graded manually**

Language: C++

Continue with your program for **Problem 12.6** in the following manner. Remember to check the mathematical validity of the parameters (you can assume that the nominator and the denominator are always numbers). Use the suggestions from slide 10 (Lecture 12) for choosing to write member methods or friend functions.

Also consider the suggestions regarding the return types and parameter types.

- Overload the operators +, − for computing the sum and difference of two fractions.
- Overload the operator = for assigning.
- Overload the operators < and > to compare two fractions.
- In your testing program you should be able to enter two fractions from the keyboard using `cin >>`. Determine the greater fraction and print it on the screen using `cout <<`. Also compute the sum and the difference of the two fractions (storing the result in other objects) and print them on the screen (one per line using the overloaded operator and `cout <<`).

In order to implement the addition and subtraction of two fractions you will have to calculate the lowest common multiple (LCM) of the denominators of the two fractions. The LCM can be computed according to the formula: $\texttt{LCM(a,b)} = \frac{\texttt{a} \cdot \texttt{b}}{\texttt{GCD(a,b)}}$, where GCD is the greatest common divisor.

Operations with fractions

Addition:

$$\frac{\texttt{a}}{\texttt{b}} + \frac{\texttt{c}}{\texttt{d}} = \frac{\texttt{a} \cdot \texttt{LCM(b,d)}/\texttt{b} + \texttt{c} \cdot \texttt{LCM(b,d)}/\texttt{d}}{\texttt{LCM(b,d)}}$$

Subtraction:

$$\frac{\texttt{a}}{\texttt{b}} - \frac{\texttt{c}}{\texttt{d}} = \frac{\texttt{a} \cdot \texttt{LCM(b,d)}/\texttt{b} - \texttt{c} \cdot \texttt{LCM(b,d)}/\texttt{d}}{\texttt{LCM(b,d)}}$$

Multiplication:

$$\frac{\texttt{a}}{\texttt{b}} \cdot \frac{\texttt{c}}{\texttt{d}} = \frac{\texttt{a} \cdot \texttt{c}}{\texttt{b} \cdot \texttt{d}}$$

Division:

$$\frac{\texttt{a}}{\texttt{b}} / \frac{\texttt{c}}{\texttt{d}} = \frac{\texttt{a} \cdot \texttt{d}}{\texttt{b} \cdot \texttt{c}}$$

Problem 12.8 *Polymorphism I*

(2 points)

Due by Monday, November 25th, 23:00 **Graded manually**

Language: C++

- Download and unzip the archive:
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/a12.zip`
  For compiling the files you can use: 1) compile the files as part of a project or 2) from the terminal using `g++`.
- Draw (using ASCII characters) a diagram how these classes relate to each other and put this into `testvirtual.cpp` as part of your comments.
- For each numbered point in the file `testvirtual.cpp` add a detailed comment about what is happening in the program.
- Like in `Circle.cpp`, output a message on the screen when the method `calcArea()` is being called in any of the classes.
- Add a method to calculate the perimeter for each class definition.
- Change the test program to additionally print the total perimeter of all objects.
- Also print a message on the screen when the method `calcPerimeter()` is being called.
- Add a `Square` class (consisting of a header file and a cpp file), and add a square object to your test program. Consider the relation of a square to the other classes.

Submit a **zip** file containing all your `.h` and `.cpp` files related to this problem.

Bonus Problem 12.9 *Polymorphism II*

(1 point)

Due by Monday, November 25th, 23:00 **Graded manually**

Language: C++

Change `testvirtual.cpp` such that 25 objects (circles, rings, rectangles, squares) are randomly created at runtime. Their colors (`RED`, `BLACK`, `VIOLET` and `BLUE`) and sizes (between 5 and 100) should also be randomly chosen.

Compute the area and the perimeter for each object and print them on the screen.
Submit a **zip** file containing all your `.h` and `.cpp` files related to this problem.

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.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a12_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.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.
- 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 Monday, November 25th, 23:00.

Assignment 11 - Dynamic Memory Allocation and Inheritance

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 11.1 *Boxes* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++

Design and write an object oriented program for the computation of the volume of boxes (having a height, a width and a depth as double values). Your solution should have methods for setting and getting the height, the width and the depth of a box as well as constructors (a default constructor and one which sets the data members), a copy constructor and a destructor.
Name your files `testbox.cpp`, `Box.cpp` and `Box.h`.
The program testing your code (`testbox.cpp`) should do the following:

a) Enter from the keyboard the a value for `n`, the number of boxes that will be entered.

b) Dynamically create an array of `2*n` boxes.

c) Enter from the keyboard the height, width and depth of each box one after the other for the first `n` boxes and the add their copies on the remainder `n` positions (in the same order). Use the copy constructor for doing this.

d) Loop across all boxes, calculate and print on the screen the volume of each box.

You can assume that the input will be valid.

Bonus Problem 11.2 *Revise the `Person` class* (1 point)

Due by Monday, November 18th, 23:00 **Graded manually**
Language: C++

Revise and adapt the `Person` class from **Bonus Problem 10.4** such that you add the following constructors: empty, parametric, copy constructor. Make sure that you use the `const` modifier properly for paramaters and methods. In the `main` function illustrate the usage of the added components.
Name the files: `Person.h`, `Person.cpp`, `testperson.cpp`.

Problem 11.3 *Inheritance with creatures* (1 point)

Due by Monday, November 18th, 23:00 **Graded manually**
Language: C++

Download the file:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/creature.cpp`
Extend the example program `creature.cpp` by adding two other type of creatures (i.e., two new classes derived from `Creature`) with some example properties and methods (use your imagination for doing this). Use/implement at least one property and at least one method for each derived class.
Each constructor, destructor and method should be implemented in such a way that each call is being documented via messages printed on the screen.
Create one instance of `Wizard`, two instances of the two other classes, and call for each object methods of the particular instances (as in the original example).
You can assume that the setting values are always valid.

Problem 11.4 *Inheritance split code* (1 point)

Due by Monday, November 18th, 23:00 **Graded manually**

Language: C++

Reorganize your previous file's (`creature.cpp`) content into three parts: declaration, implementation of the classes and your test program. Name the files `Creature.h`, `Creature.cpp` and `testcreature.cpp`.

Problem 11.5 *Inheritance and pointers* (1 point)

Due by Monday, November 18th, 23:00 **Graded manually**

Language: C++

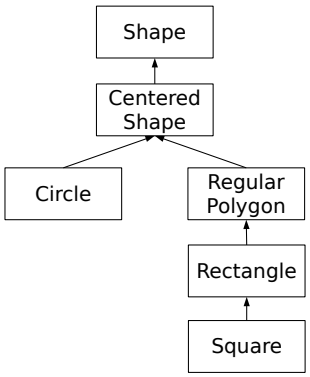
Change your previous testing program (`testcreature.cpp`) such that it runs in an endless loop and waits for input from the keyboard. If the word "`wizard`", "`object1`", or "`object2`" is entered, a wizard, your other `object1` or your other `object2` should be dynamically created (via `new`), the corresponding method is being called and the object is then destroyed (via `delete`). Entering "`quit`" should stop the execution of the program.
Name the files `Creature.h` (same as above), `Creature.cpp` (same as above) and `dyncreature.cpp`.
You can assume that the input will be valid.

Problem 11.6 *Shapes* (1 point)

Due by Monday, November 18th, 23:00 **Graded manually**

Language: C++

Download the files:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.h`
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.cpp`
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/testshapes.cpp`
Extend the `Shapes` example by adding two new classes `Rectangle` and `Square` according to the following inheritance diagram.



Implement the following:

a) default constructors for all classes by initializing the properties to default values,

b) setters and getters for all classes,

c) the constructors `Rectangle(const string& n, double nx, double ny, double nwidth, double nheight)` and `Square(const string& n, double nx, double ny, double nside)`,

d) copy constructors for all classes,

e) the methods `double perimeter()` and `double area()` for the classes `Circle`, `Rectangle` and `Square` for returning the perimeter and area of corresponding instances,

f) in your testing program create instances of `Circle`, `Rectangle` and `Square`, and then compute and print on the screen their perimeter and area.

Name your files `Shapes.h`, `Shapes.cpp` and `testshapes.cpp`.
You can assume that the setting values are always valid.

Bonus Problem 11.7 *A `Vector` class for doubles* (2 points)

Due by Monday, November 18th, 23:00 **Graded manually**

Language: C++

Create a class named `Vector` for storing and managing vectors of doubles. The properties of a vector are the size of the vector and a double pointer pointing to a memory location where the components of the vector are stored. The class has to provide a default constructor, another constructor for setting the properties, a copy constructor and a destructor. The non-default constructors should dynamically allocate memory for a vector and the destructor should release it. Provide suitable setter and getter methods for each property and a method for printing the components of a vector on the screen separated by spaces. Also provide methods for computing the norm of a vector, the sum, the difference and the scalar product of two vectors. The class declaration has to be placed into `Vector.h`, the class definition has to be placed into `Vector.cpp` and the test program where the instances are created has to be in `testvector.cpp`. The test program should create three instances of the `Vector` class using the different constructors (the second instance should be the copy of the first one). Then the norm of the first vector should be computed and printed on the screen, the scalar product, the sum and the difference of the first and third vectors should be also computed and printed on the screen.

Use the `const` modifier properly for parameters and methods.
You can assume that the input or the setting values are valid.
The prototypes should have the following form:

```
double norm();
Vector add(const Vector) const;
```

The usage should be:

```
Vector v1, v2;
(v1.add(v2)).print();
```

Note: If $v_1 = (a_1, a_2, \ldots, a_n)$ and $v_2 = (b_1, b_2, \ldots, b_n)$ with $n \in \mathbb{N}^*$. Then:

$\|v_1\| = \sqrt{\sum_{i=1}^{n} a_i^2}$ (the norm of a vector),

$v_1 \cdot v_2 = \sum_{i=1}^{n} a_i \cdot b_i$ (the scalar product of two vectors),

$v_1 + v_2 = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$, and

$v_1 - v_2 = (a_1 - b_1, a_2 - b_2, \ldots, a_n - b_n)$.

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.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a11_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at **`https://cantaloupe.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.
- 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 Monday, November 18th, 23:00.

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

Course: CH-230-A
November 5th, 2019

Assignment 10 - Makefiles and Function Pointers

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 10.1 *Error messages produced by the compiler* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++
Download and use the files:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Critter.h`
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Critter.cpp` and
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/testcritter.cpp`.
You will need to create a project to successfully compile all three files (or compile them from the command line as specified on the slides).

a) Comment out the `using namespace std;` and then take your time, read and interpret the error messages.

b) Also remove the `Critter::` prefix in one of the methods in `Critter.cpp`, read and interpret the error message.

Then create a file called `explanations.txt`. This file should be uploaded together with the other files and should contain your descriptions and interpretations of the errors as well as your comments on potential alternative solutions.
You can assume that the input will be valid.

Problem 10.2 *The `Critter` class* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++
Use the previously given files: `Critter.h`, `Critter.cpp` and `testcritter.cpp`.
Expand `Critter.h` by two additional properties of your choice, and corresponding setter and getter methods, then adjust `Critter.cpp` and `testcritter.cpp` accordingly.
Also adapt the `print()` method such that the new properties are printed on the screen as well.
You can assume that the input will be valid.

Problem 10.3 *A `City` class* (1 point)

Due by Monday, November 11th, 23:00 **Graded manually**
Language: C++
Create a class named `City`. Assume that a city has a name, a number of inhabitants, a mayor and an area (in km^2).
Then create three instances of this class: *Bremen*, *Paris* and *London*. Provide suitable setter and getter methods for each of these properties. The class declaration has to be placed into `City.h`, the class definition has to be placed into `City.cpp` and the test program where the instances are created has to be in `testcity.cpp`.
You can set the needed data from the `main()` function by initialization or read it from the keyboard.

Bonus Problem 10.4 *A `Person` class* (1 point)

Due by Monday, November 11th, 23:00 **Graded manually**

Language: C++

Create a class named `Person`. Add at least four properties/attributes to the class which should not all have the same type. The properties should be `private`.
Then create three instances of this class. Provide suitable setter and getter methods for each of these properties. The class declaration has to be placed into `Person.h`, the class definition has to be placed into `Person.cpp` and the test program where the instances are created has to be in `testperson.cpp`.
You can set the needed data from the `main()` function by initialization or read it from the keyboard.

Problem 10.5 *Constructors for `Critter`* (1 point)

Due by Monday, November 11th, 23:00 **Graded manually**

Language: C++

Add three constructors to the class `Critter`. Each constructor should also print a simple informational message on the screen such that one can see when and which constructor has been called.
You should be able to create an instance of the `Critter` class
(1) without supplying any properties (which should set the name to "`default_critter`", the height to 5 and the rest to 0),
(2) by only supplying a name as parameter (which should set the height to 5 and the rest to 0), and also
(3) by supplying *name*, *hunger*, *boredom* and *height* all as parameters. You should also be able to create an instance of the `Critter` class without specifying the *height*. If the *height* is not supplied, the critter has the default height of 10.
Write a test program which creates four instances of the `Critter` by using these three different constructors (the last one in two ways). Set their *hunger* levels to 2 by using appropriate method and/or constructor calls. The critters' properties should then be printed on the screen.
Name the files `Critter.h`, `Critter.cpp` and `testcritter.cpp`.

Problem 10.6 *Information hiding I* (1 point)

Due by Monday, November 11th, 23:00 **Graded manually**

Language: C++

A game developer crew has decided to rather use a percentage scale (double value between 0.0 and 1.0) to represent the *hunger* level of a critter. Change the internal structure of the class to reflect this. However, your **existing test program should run without any modifications** (therefore the public class interface stays the same) and you will need to find a way to convert the current *hunger* levels from integer values (which are from 0 to 10) to doubles and then from doubles back to integers. Use separate methods for doing this.
Use a simple mapping scheme like 10 is 1.0, 9 is 0.9, 8 is $0.8 \ldots 1$ is 0.1, and 0 is 0.0.
Name the files `Critter.h`, `Critter.cpp` and `testcritter.cpp` (**must remain unchanged**).
The implementation for the conversions needs to be put into `Critter.cpp` and should not be part of the public interface.
The client program `testcritter.cpp` from the previous problem **must remain unchanged**.
The *hunger* levels of the critters should be "internally" at 0.2 (meaning 20%).
You can assume that the setting values are always valid.

Problem 10.7 *Information hiding II* (1 point)

Due by Monday, November 11th, 23:00 **Graded manually**

Language: C++

Next a *thirst* level (as double value) should be added to the properties of a critter. Add a new constructor that takes five parameters for setting all properties of a critter.
Make also sure that the existing constructors will still work. For the existing constructors, the *thirst* level should be set to the same level as the *hunger* level. Your existing `testcritter.cpp` must still be able to run **in its unchanged form**. So the already existing constructors need to support the change. Name the files `Critter.h`, `Critter.cpp` and `testcritter.cpp`.
Finally, you should adapt the print method for printing on the screen also the value of the thirst level as a double. The client program `testcritter.cpp` may contain one additional line, where the constructor taking five parameters is being called.
You can assume that the setting values are always valid.

Problem 10.8 *Copy constructor* (1 point)

Due by Monday, November 11th, 23:00 **Graded manually**

Language: C++

Download the file:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/copyconstructor.cpp`

Based on the source code of `copyconstructor.cpp` implement the method `funcByref()`. Change all constructors (including the copy constructor) such that **you can clearly see when and which of them is invoked** by adding a message which is printed on the screen.

Then in your `main()` function create at least two objects using the different constructors, call `funcByVal()`, `funcByRef()`, and print the results on the screen. Then make sure that the memory occupied by the objects will be released by the end of the program.

You can assume that the setting values are always valid.

Problem 10.9 *A `Complex` class* (2 points)

Due by Monday, November 11th, 23:00 **Graded manually**

Language: C++

Create a class named `Complex` for storing and managing complex numbers. A complex number has an real part and an imaginary part. The class has to provide a default constructor initializing the properties by 0, another constructor for setting the properties with specific values, a copy constructor and an empty destructor. Provide suitable setter and getter methods for each property and a method for printing the complex number on the screen in its mathematical form (e.g., $1 + 2i$, $3 - 5i$). Also provide methods for the conjugation of a complex number, and for adding, subtracting and multiplying two complex numbers. The class declaration has to be placed into `Complex.h`, the class definition has to be placed into `Complex.cpp` and the test program where the instances are created has to be in `testcomplex.cpp`. The test program should create at least two instances of the `Complex` class, the data for the properties should be read from the keyboard. Then:

a) the conjugate of the first instance should be determined and printed on the screen;

b) the sum of the two instances should be determined and printed on the screen;

c) the difference between the second and first instance should be determined and printed on the screen;

d) the multiplication of the two instances should be determined and printed on the screen.

The prototypes of the methods for adding, subtracting and multiplying must have the following form:

```
Complex Complex::add(Complex);
```

Then the usage will be the following:

```
Complex c1, c2, c3;
...
c3 = c1.add(c2);
```

Note: If $z = a + bi$ then $\overline{z} = a - bi$. If $z_1 = a + bi$ and $z_2 = c + di$ then $z_1 + z_2 = (a + c) + (b + d)i$, $z_1 - z_2 = (a - c) + (b - d)i$ and $z_1 \cdot z_2 = (a \cdot c - b \cdot d) + (b \cdot c + a \cdot d)i$.

You can assume that the input will be valid.

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.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a10_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.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.
- 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 Monday, November 11th, 23:00.

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

Course: CH-230-A
October 29th, 2019

Assignment 9 - C++ Introduction, Function Overloading, References, Dynamic Memory Allocation

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules: https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 9.1 *Check g++ compiler & Write simple program* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++

Make sure that you have `g++` installed and you can use an IDE or an editor to write and compile C++ code.
Write a program that reads your country of origin from the standard input (i.e., keyboard) and prints it on the standard output (i.e., screen) using `cin` and `cout`.
You can assume that the input will be valid and will not contain spaces.

Problem 9.2 *Using different variables* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++

Write a program which reads one integer value `n`, one double value `x` and a string `s` from the keyboard. Then `s` and `x` should be printed on the screen (separated by ':' with a newline after the double value). This printing should be repeated `n` times.
You can assume that the input will be valid and the string will not contain spaces.

Problem 9.3 *Absolute value function* (1 point)

Due by Monday, November 4th, 23:00 **Graded manually**
Language: C++

Consider the `abs` function, which determines the absolute values of a real number. The function returns the following values:

$$\texttt{abs(x)} = \begin{cases} -\texttt{x} & \text{for} \quad \texttt{x} < 0, \\ \texttt{x} & \text{for} \quad \texttt{x} \geq 0. \end{cases}$$

Write a function which determines and returns the absolute values of a float parameter. Then write a `main()` function which calls the function from above and prints on the screen the returned value. You may not use any library functions related to the absolute value.
You can assume that the input will be valid.

Problem 9.4 *Function overloading* (1 point)

Due by Monday, November 4th, 23:00 **Graded manually**
Language: C++

Write a program that provides two overloaded functions named `... mycount(...)`. This function either computes the difference between the second and first parameter (in this order) if integers are passed or counts the number of occurrences of a character if a character and a string are passed.
For example, `mycount(7, 3)` should return -4 and `mycount('i', "this is a string")` should return 3. In case of no occurrence 0 should be returned.
Write a simple `main()` function that demonstrates the above described behavior for both functions.
You can assume that the input will be valid.

Bonus Problem 9.5 *Concatenating loop* (1 point)

Due by Monday, November 4th, 23:00 **Graded manually**

Language: C++

Write a program which reads strings one by one from the keyboard until the string `"quit"` is entered. Use a loop with and `bool` variable for exiting the loop. At the end a concatenated string containing all entered strings (without spaces or other characters in between) should be printed on the screen (except the `"quit"`).

You can assume that the input will be valid and the entered strings may contain spaces.

Problem 9.6 *A guessing game* (2 points)

Due by Monday, November 4th, 23:00 **Graded manually**

Language: C++

Write a simple program for implementing the guessing game as outlined in the slides (Tutorial 9, slides $23 - 24$).

You can assume that the input will be valid.

Problem 9.7 *Function overloading* (1 point)

Due by Monday, November 4th, 23:00 **Graded manually**

Language: C++

Write three overloaded functions `... myfirst(...)` which should do the following:

1) if called with an array of integers, it determines and returns the first positive and even value from the array. If no such element exists then -1 should be returned;
2) if called with an array of doubles, it determines and returns the first negative element which does not have a fractional part. If no such element exists then -1.1 should be returned;
3) if called with an array of chars, it determines and returns the first element which is a consonant. If no consonants are present in the array then the character '0' should be returned.

Write a program which calls the above functions and illustrates their effect. You may choose to enter test data from the keyboard or to initialize variables within your code.

You can assume that the input will be valid.

Problem 9.8 *Swapping with call-by-reference* (1 point)

Due by Monday, November 4th, 23:00 **Graded manually**

Language: C++

Write a program `swapref.cpp`, which provides three overloaded functions `void swapping(...)`. These functions should swap two integers, two floats, and two pointers to char. The swapping should be done by a "real" `call-by-reference` (i.e., not by using `*`). Complete the following code fragment:

```
#include <iostream>

using namespace std;

void swapping(...) { .... } // swap ints
void swapping(...) { .... } // swap floats
void swapping(...) { .... } // swap char pointers

int main(void) {
   int a = 7, b = 15;
   float x = 3.5, y = 9.2;
   const char *str1 = "One";
   const char *str2 = "Two";

   cout << "a=" << a << ", b=" << b << endl;
   cout << "x=" << x << ", y=" << y << endl;
   cout << "str1=" << str1 << ", str2=" << str2 << endl;

   swapping(a, b);
   swapping(x, y);
   swapping(str1, str2);
```

```
    cout << "a=" << a << ", b=" << b << endl;
    cout << "x=" << x << ", y=" << y << endl;
    cout << "str1=" << str1 << ", str2=" << str2 << endl;
    return 0;
}
```

Problem 9.9 *Dynamic allocation and references* (1 point)

Due by Monday, November 4th, 23:00 **Graded manually**

Language: C++

Write a program which reads from the keyboard an integer `n` followed by `n` integer values which are to be stored in a dynamically allocated array `a`. Your program should define a function `void subtract_max(...)` for determining the maximum value in the array and subtracting this maximum from all elements of the array. You should also define a function called `void deallocate(...)` for releasing the memory which was allocated for the array. The `main()` function should allocate memory for the array, call the function, illustrate its effect by printing the values of the array and finally deallocate the memory occupied by the array by calling the second function from above.

You can assume that the input will be valid.

Problem 9.10 *Word guessing* (2 points)

Due by Monday, November 4th, 23:00 **Graded manually**

Language: C++

Write a program that stores an array of words (containing "`computer`", "`television`", "`keyboard`", "`laptop`", "`mouse`") and 12 other words of your choice in an array of strings. Inside of a game loop your program should randomly choose one word out of the 17 possible words. The program should print the word on the screen after replacing all vowels by underscores, then the player should try to guess the word. After the player has guessed the right word, the number of tries should be printed on the screen and the player should be asked whether he/she wishes to play again. If the player enters "`quit`" as a word guess, then the game should immediately stop.

You can assume that the input will be valid and that "`quit`" will not be in the set of words to be guessed.

Bonus Problem 9.11 *Checking words for palindrome* (1 point)

Due by Monday, November 4th, 23:00 **Graded manually**

Language: C++

A palindrome is a word that is the same read forward and backwards (e.g., "`level`", "`ana`", "`civic`", etc.). Write a function `bool isPalindrome(string s)` that recognizes this characteristic of a string by returning `true` or `false`. Your program should have the form of a game as the previous program with the difference that the player enters a word and then the function should be called to check if the word is a palindrome or not and a corresponding message should be printed on the screen. This should be repeated until "`exit`" is entered as a word (i.e., then the game should immediately stop).

You can assume that the input will be valid.

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.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a9_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.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.
- 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 Monday, November 4th, 23:00.

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

Course: CH-230-A
October 22nd, 2019

Assignment 8 - Queues, Stacks, Files

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
  https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 8.1 *Adding to the queue* (2 points)

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

Download the files:

```
https://grader.eecs.jacobs-university.de/courses/320112/c/queue.h
https://grader.eecs.jacobs-university.de/courses/320112/c/queue.c
https://grader.eecs.jacobs-university.de/courses/320112/c/testqueue.c
```

Take a look at the three files and understand the source code. Extend the code of `queue.c` by implementing the `enqueue()` function. Follow the hints given in the slides (see Tutorial 8, slide 12).
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 8.1: input

```
a
3
a
5
a
7
q
```

Testcase 8.1: output

```
add int: Putting 3 into queue
1 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 5 into queue
2 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 7 into queue
3 items in queue
Type a to add, d to delete, q to quit:
Bye.
```

Problem 8.2 *Removing from the queue* (2 points)

Due by Monday, October 28th, 23:00 **Graded manually**
Language: C

Extend the source code of `queue.c` from **Problem 8.1** by implementing the `dequeue()` function. Follow the hints given in the slides (see Tutorial 8, slide 13) and consider the case of a queue underflow.
You can assume that the input will be valid except the semantical possibility of reaching queue underflow. To pass the testcases your output has to be identical with the provided ones.

Testcase 8.2: input

```
a
3
a
5
a
7
d
d
q
```

Testcase 8.2: output

```
add int: Putting 3 into queue
1 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 5 into queue
2 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 7 into queue
3 items in queue
Type a to add, d to delete, q to quit:
Removing 3 from queue
2 items in queue
Type a to add, d to delete, q to quit:
Removing 5 from queue
1 items in queue
Type a to add, d to delete, q to quit:
Bye.
```

Problem 8.3 *Printing the queue* (2 points)

Due by Monday, October 28th, 23:00 **Graded automatically with testcases only**

Language: C

Extend the source code of `queue.h`, `queue.c` and `testqueue.c` from **Problem 8.2** by adding and implementing the additional function `printq()` for printing the elements of the queue separated by spaces. If you enter '`p`', then the program should print the elements of the queue. Make sure that you can print more than once.

You can assume that the input will be correct. To pass the testcases your output has to be identical with the provided ones.

Testcase 8.3: input

```
a
3
a
5
a
7
p
q
```

Testcase 8.3: output

```
add int: Putting 3 into queue
1 items in queue
Type a to add, d to delete, p to print, q to quit:
add int: Putting 5 into queue
2 items in queue
Type a to add, d to delete, p to print, q to quit:
add int: Putting 7 into queue
3 items in queue
Type a to add, d to delete, p to print, q to quit:
content of the queue: 3 5 7
3 items in queue
Type a to add, d to delete, p to print, q to quit:
Bye.
```

Problem 8.4 *A stack for converting numbers* (2 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Modify the stack implemented for **Problem 7.7** such that you can use it for converting a positive decimal number stored in an `unsigned int` into the binary representation of the number using division by 2 and storing the remainder of the division by 2 in the stack.

Upload again all files related to this problem (i.e., `stack.h`, `stack.c` and `convertingstack.c`).

You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 8.4: input

```
75
```

Testcase 8.4: output

```
The binary representation of 75 is 1001011.
```

Bonus Problem 8.5 *A stack of words* (3 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Modify the `struct` from **Problem 8.4** and write a program that tests a stack of words (the words will not be longer than 30 characters). Keep in mind the functions `strcpy()`, `strcmp()` and `strcat()`.

Use the word stack to check if a sentence (assume that the words are separated by spaces, all letters are lowercase and no punctuation marks are contained) is palindromic by words. For example, the sentence "`dogs like cats and cats like dogs`" is palindromic by words, because it reads the same from backwards (word by word). The program should terminate its execution if "`exit`" is entered as a sentence.

Your program should consist of one header file and two `.c` files (i.e., `stack.h`, `stack.c` and `wordstack.c`).

You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 8.5: input

```
dogs like cats and cats like dogs
bob likes tomatos do not like bob
exit
```

Testcase 8.5: output

```
The sentence is palindromic by words!
The sentence is not palindromic by words!
```

Problem 8.6 *Read chars and write an int* (1 point)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads the first two characters from the file "`chars.txt`" and writes the sum of their ASCII code values as a number into "`codesum.txt`". Use an editor to create the input file "`chars.txt`". Your program is responsible to create the output file "`codesum.txt`".

You can safely assume that the content of the input file will be valid.

Problem 8.7 *Read and write doubles* (1 point)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads from the keyboard the names of two files containing two double numbers. Your program should read these two values from the two files, compute their sum, difference, product and division, and write the results on separate lines into the file "`results.txt`".

You can safely assume that the input is valid, the two input files exist and each contains one valid double value.

Bonus Problem 8.8 *Merge two files* (1 point)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads the content of two files "`text1.txt`" and "`text2.txt`" line by line and merges them into another file called "`merge12.txt`".

You can safely assume that the input is valid.

Problem 8.9 *Counting words in a file* (2 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads the content of a file given as input and counts the number of the words in the file. It is assumed the words are separated by one or multiple of the following characters: `' ' ','` `'?'` `'!'` `'.'` `'\t'` `'\r'` `'\n'`.

For testing your solution to this problem, please use:

`https://grader.eecs.jacobs-university.de/courses/320112/c/words.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/words2.txt`

You can assume that the content of the input file will be valid if existing.

Testcase 8.9: input

```
words.txt
```

Testcase 8.9: output

```
The file contains 17 words.
```

Bonus Problem 8.10 *"Authentification" with files* (3 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads from the standard input a username and a password, and prints a message on the screen for granting or denying access depending on the fact if the user and the corresponding password are listed in a file which is given as input to the program.

The program should repeatedly check the username and its password introduced from the standard input until the word "`exit`" is introduced for the username. It is assumed that the word "`exit`" is not contained in the input file.

Do not store the whole information (list of usernames and passwords) in the main memory or do not read the whole information from the file for every request, but store only partial information (e.g., the usernames) and use the functions `ftell()` and `fseek()` to read the rest of the needed information.

For testing your solution to this problem, please use:

`https://grader.eecs.jacobs-university.de/courses/320112/c/users.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/users2.txt`

You can assume that the content of the input file will be valid if existing.

Testcase 8.10: input

```
users.txt
ben
12b43
kate
jfd45
bill
iu3556
exit
```

Testcase 8.10: output

```
Access to user ben is granted.
Access to user kate is denied.
Access to user bill is granted.
Exiting ...
```

Problem 8.11 *Concat `n` files* (3 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads from the standard input the value of an integer `n` and then the names of `n` files. The program should concatenate the content of the `n` files separated by `'\n'` and write the result on the standard output and also into `output.txt`.

Read the input files and write the output file using the binary mode. Use a char buffer of size 64 bytes and chunks of size 1 byte when reading and the same buffer with chunks of size 64 bytes (or less if the last write and file size is not a multiply of 64) when writing.

For testing your solution to this problem, please use:

`https://grader.eecs.jacobs-university.de/courses/320112/c/file1.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/file2.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/file3.txt`

You can assume that the content of the input files will be valid if existing.

Testcase 8.11: input

```
3
file1.txt
file2.txt
file3.txt
```

Testcase 8.11: output

```
Concating the content of 3 files ...
The result is:
The first file's
content.
The second file's content.
The
third files's
content.
The result was written into output.txt
```

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.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a8_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.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.
- 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 Monday, October 28th, 23:00.

Assignment 7 - Multiple Sources, Linked Lists, Function Pointers, Stacks

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules: https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 7.1 *Multiple sources* (1 point)

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

Modify your solution for **Problem 6.8** such that you separate your source code into three files: struct declaration and function declarations in `linked_list.h`, function definitions in `linked_list.c`, and your `main` function in `use_linked_list.c`. Use exactly the provided names for your files.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.1: input

```
b
2
b
3
a
4
p
r
p
q
```

Testcase 7.1: output

```
3 2 4
2 4
```

Problem 7.2 *A doubly linked list of characters* (4 points)

Due by Monday, October 21st, 23:00 **Graded manually**
Language: C

Create a data type that implements a doubly linked list of characters. Write a program that tests your double linked list with the testcase given below. An integer value `1` entered from the keyboard will add the following character to the list to the beginning of the list, while a `2` will remove all elements with the given character from the list, a `3` will print the current list, while a `4` will print the elements of the list backwards. A `5` will empty the list, free the memory used by the doubly linked list and quit the execution of the program.
You can assume that the input does not contain logical errors (i.e., if the command is `2`, you can assume that a character will follow). However, a character to be deleted may not be currently in the list. In this case, the message "`The element is not in the list!`" should be printed on the standard output.
Use a switch-case statement to decide which action to take.
You can assume that the input will be valid regarding the structure. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.2: input

```
1
r
1
a
1
d
3
1
a
1
x
1
r
1
x
3
2
x
3
4
2
b
5
```

Testcase 7.2: output

```
d a r
x r x a d a r
r a d a r
r a d a r
The element is not in the list!
```

Problem 7.3 *Makefile* (2 points)

Due by Monday, October 21st, 23:00 **Graded manually**
Language: C

Continue with your solution for **Problem 7.1** in the following manner: write and upload a makefile called "`Makefile.txt`" which has multiple targets and can be used to: 1) generate all object files corresponding to the previous source files, 2) generate executable code from the object files and 3) delete all generated object files and the executable.
Submit the three source files and the makefile called "`Makefile.txt`".
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Problem 7.4 *Simple function pointers* (2 points)

Due by Monday, October 21st, 23:00 **Graded automatically with testcases only**
Language: C

Write a program that reads a string and then repeatedly reads a command (one character) from the standard input.
If you press '`1`', then the string is printed uppercase on the standard output.
If you press '`2`', then the string is printed lowercase on the standard output.
If you press '`3`', then lowercase characters are printed uppercase and uppercase characters are printed lowercase on the standard output.
If you press '`4`', then the program should quit the execution.
Your `main` function (where you read the commands) or your other functions may not contain any `if` or `switch` statements for mapping the command to which function should be called. Your `main` function should contain an endless `while` loop.
Implement the solution using a function pointer array. The original string should not be changed.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.4: input

```
This is a String
1
2
3
2
1
4
```

Testcase 7.4: output

```
THIS IS A STRING
this is a string
tHIS IS A sTRING
this is a string
THIS IS A STRING
```

Problem 7.5 *Quicksort with function pointers* (2 points)

Due by Monday, October 21st, 23:00 **Graded automatically with testcases only**

Language: C

Write a program that sorts an array of `n` integers. After reading `n` and the values of the array from the standard input, the program reads a character and if this character is `'a'` then the sorting should be ascending, if the character is `'d'` then the sorting should be descending and if the character is `'e'` then the program should quit execution.

Your `main` function should contain an endless `while` loop for getting repeated input.

Your program should use function pointers and for sorting you should use the function `qsort` from `stdlib.h`.

You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.5: input

```
5
2
4
1
5
3
d
a
e
```

Testcase 7.5: output

```
5 4 3 2 1
1 2 3 4 5
```

Problem 7.6 *Bubblesort with function pointers* (4 points)

Due by Monday, October 21st, 23:00 **Graded manually**

Language: C

Write a program that reads an array of the following structure and sorts the data in ascending order by name or age using the *bubblesort algorithm*.

```
struct person {
        char name[30];
        int age;
};
```

Your program should read the number of persons from the standard input followed by the array of data corresponding to the persons. You should print the lists of sorted persons in ascending order with respect to their name (alphabetical order) and with respect to their age. Within the sorting according to age, note that if multiple persons have the same age, then they should be sorted alphabetically with respect to their name. Within the sorting according to name, note that if multiple persons have the same name, then they should be sorted with respect to their age.

Instead of writing two sorting functions use function pointers such that you can implement one `bubblesort` function able to sort according to different criteria.

You can assume that the input will be valid and that the names will not contain spaces. To pass the testcases your output has to be identical with the provided ones.

The pseudocode of the bubblesort algorithm is the following:

```
repeat
  swapped = false
  for i = 1 to length(A) - 1 inclusive do:
  /* if this pair is out of order */
    if A[i-1] > A[i] then
      /* swap them and remember something changed */
      swap( A[i-1], A[i] )
      swapped = true
    end if
  end for
until not swapped
```

Testcase 7.6: input

```
3
anne
23
mary
18
bob
20
```

Testcase 7.6: output

```
{anne, 23}; {bob, 20}; {mary, 18};
{mary, 18}; {bob, 20}; {anne, 23};
```

Problem 7.7 *A stack of integers* (3 points)

Due by Monday, October 21st, 23:00 **Graded automatically with testcases only**

Language: C

Implement a stack, which is able to hold maximum 12 integers using an array implementation. You need to implement the functions `... push(...)`, `... pop(...)`, `... empty(...)` for emptying the stack, and `... isEmpty(...)` for checking if the stack is empty.
Your program should consist of `stack.h` (`struct` definition and function declarations), `stack.c` (function definitions) and `teststack.c` (`main` function) and should use the following `struct`:

```
struct stack {
  unsigned int count;
  int array[12];        // Container
};
```

Input structure

There are several commands that manipulate the stack.
These commands are:

- `s` followed by a number pushes the number into the stack,
- `p` pops a number on the top off the stack and prints it on the standard output,
- `e` empties the stack by popping one element after the other and printing them on the standard output,
- `q` quits the execution of the program.

Output structure

If an element is popped off the stack then the element is printed. Stack underflow and overflow should be detected and an informational message (either "`Stack Overflow`" or "`Stack Underflow`" should be printed on the screen. In these cases no operation takes place.

You can assume that the input will be correct. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.7: input

```
s
5
s
7
p
s
3
e
p
q
```

Testcase 7.7: output

```
Pushing 5
Pushing 7
Popping 7
Pushing 3
Emptying Stack 3 5
Popping Stack Underflow
Quit
```

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.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a7_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.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.
- 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 Monday, October 21st, 23:00.

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

Course: CH-230-A
October 8th, 2019

Assignment 6 - C Preprocessor, Bitwise Operators, Linked Lists

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules: https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 6.1 *Swapping two variables* (2 points)

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

Write a macro and a program for swapping the contents of two variables. The macro should have three parameters: the two variables and the corresponding data type.
Your program should read two integers and two floats from the standard input. Then you should print on the standard output the contents of the four variables after swapping (floats with a floating point precision of 6).
You can 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 6.1: input

```
1
2
3.45
5.677
```

Testcase 6.1: output

```
After swapping:
2
1
5.677000
3.450000
```

Problem 6.2 *Determine the least significant bit* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C

Write a macro and a program for determining the least significant bit (the first bit from the right in the binary representation) of an `unsigned char` read from the standard input.
Your program should read an `unsigned char` from the standard input and print the decimal representation of the `unsigned char` as well as its least significant bit (which is either 1 or 0) on the standard output using only bitwise operators and without explicitly converting to binary representation.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 6.2: input

```
F
```

Testcase 6.2: output

```
The decimal representation is: 70
The least significant bit is: 0
```

Problem 6.3 *Determine the mid-range of three values* (2 points)

Due by Monday, October 14th, 23:00 **Graded automatically with testcases only**

Write multiple macros and a program for determining the mid-range of three values. The mid-range of three variables `a`, `b`, and `c` is calculated as

$$mid\_range(a,b,c) = \frac{\min(a,b,c) + \max(a,b,c)}{2}.$$

For example if 3, 10, 1 is the input, the mid-range of these values is

$$mid\_range(3,10,1) = \frac{\min(3,10,1) + \max(3,10,1)}{2} = \frac{1+10}{2} = \frac{11}{2} = 5.5.$$

Your program should read three integers from the standard input. For calculating the mid-range of these values only macros should be used. The mid-range should be printed on the standard output with a floating point precision of 6.
You can 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 6.3: input

```
3
10
1
```

Testcase 6.3: output

```
The mid-range is: 5.500000
```

Problem 6.4 *Conditional compilation for showing intermediate results* (2 points)

Due by Monday, October 14th, 23:00 **Graded manually**
Language: C

Write a program which computes the scalar product of two n-dimensional integer vectors and uses conditional compilation for showing/not showing intermediate results (products of the corresponding components). The scalar product of two n-dimensional vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ is calculated as

$$< x, y >= \sum_{i=1}^{n} x_i \cdot y_i.$$

For example the scalar product of the vector $x = (1, 2, 3)$ with the vector $y = (3, 5, 1)$ is

$$< x, y >= 1 \cdot 3 + 2 \cdot 5 + 3 \cdot 1 = 3 + 10 + 3 = 16.$$

The intermediate results which are to be shown or not are 3, 10 and 3.
Your program should read from the standard input the dimension of the vector (in the previous example 3) along with the components of two integer vectors. The output consists of the intermediate results and the value of the scalar product of the two vector if the directive INTERMEDIATE is defined. If INTERMEDIATE is not defined then only the scalar product of the two vectors should be printed on the standard output.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 6.4: input

```
3
1
2
3
3
5
1
```

Testcase 6.4: output

```
The intermediate product values are:
3
10
3
The scalar product is: 16
```

Problem 6.5 *Binary representation backwards* (1 point)

Due by Monday, October 14th, 23:00 **Graded automatically with testcases only**
Language: C

Write a program using bit masks and bitwise operators for printing the binary representation of an `unsigned char` backwards. For example the character '2' is encoded as 50 in decimal representation which is in binary representation 110010. Therefore, the backwards binary representation is 010011.

Your program should read an `unsigned char` from the standard input and print on the standard output the backwards binary representation of the read character without explicitly converting the decimal value to binary or using an array to store the bits.
You can 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 6.5: input

```
2
```

Testcase 6.5: output

```
The decimal representation is: 50
The backwards binary representation is: 010011
```

Problem 6.6 *Binary representation* (2 points)

Due by Monday, October 14th, 23:00 **Graded manually**
Language: C

Write a program using bit masks and bitwise operators for printing the binary representation of an `unsigned char` without storing the bits in an array or explicitly converting to binary. For example the character `'2'` is encoded as 50 in decimal representation which is in binary representation on 8 bits 00110010.
Your program should read an `unsigned char` from the standard input and print on the standard output the binary representation of the read character.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 6.6: input

```
2
```

Testcase 6.6: output

```
The decimal representation is: 50
The binary representation is: 00110010
```

Problem 6.7 *`set3bits()`* (2 points)

Due by Monday, October 14th, 23:00 **Graded automatically with testcases only**
Language: C

Write a program for setting three bits of an `unsigned char` to 1. The function `set3bits` should have four parameters: the `unsigned char` to be changed and the three bits which are to be set to 1. For example the character `'2'` is encoded as 50 in decimal representation which is in binary representation on 8 bits 00110010. If `set3bits()` with bits 7, 6 and 1 to be set to 1 is called then the output on the standard output should be 11110010. Print the result on the standard output from the `main()` function.
You can 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 6.7: input

```
2
7
6
1
```

Testcase 6.7: output

```
The decimal representation is: 50
The binary representation is: 00110010
After setting the bits: 11110010
```

Problem 6.8 *A linked list* (1 point)

Due by Monday, October 14th, 23:00 **Graded automatically with testcases only**
Language: C

Using the example from the slides (Lecture 6, pages $19 - 24$), write a program that uses a linked list. Your program should wait for input from the keyboard. Entering from the keyboard an `'a'` will just add the following number (read as next from the keyboard) to the end of the list, while a `'b'` inserts at the beginning of the list. The character `'r'` will remove the first element from the list, a `'p'` will print the list while a `'q'` will free the memory used by the list and quit the execution of the program.

Use a switch-case statement to decide which action to take.
You can assume that the input will be valid regarding the structure. To pass the testcases your output has to be identical with the provided ones.

Testcase 6.8: input

```
b
2
b
3
a
4
p
r
p
q
```

Testcase 6.8: output

```
3 2 4
2 4
```

Problem 6.9 *An enhanced linked list* (2 points)

Due by Monday, October 14th, 23:00 **Graded manually**

Language: C

Extend your program for **Problem 6.8** by writing a function for inserting a new element into the list at a given position and a function for reversing the order of the elements in the list. Your program should wait for input from the keyboard. An `'i'` followed by two numbers (the position and the number to be inserted) should insert the second the number at position of the first number (the first element in the list has position 0). You can assume that the input does not contain any logical errors (e.g., `'i'` is always followed by two numbers, and `'b'` and `'a'` are followed by one number). However, if the position for inserting is negative or is greater than the number of elements in the list then print on the standard output "`Invalid position!`". An `'R'` should reverse the order of the elements in the list without allocating new nodes or using a doubly linked list (i.e., only with the use of pointers).
Use a switch-case statement to decide which action to take.
You can assume that the input will be valid regarding the structure. To pass the testcases your output has to be identical with the provided ones.

Testcase 6.9: input

```
b
2
b
3
a
4
p
r
p
i
1
5
p
i
4
11
R
p
q
```

Testcase 6.9: output

```
3 2 4
2 4
2 5 4
Invalid position!
4 5 2
```

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
   a6_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, October 14th, 23:00.

Assignment 5 - Arrays, Dynamic Memory Allocation, Recursion

- 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 5.1 *Triangle chars* (1 point)

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

Write a function that takes two arguments: an integer `n` and a character `ch`. The function should print the character `ch` in a triangle form as below.
Write a simple program that reads the appropriate variables and prints the result to screen by calling the function.
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 5.1: input

```
4
$
```

Testcase 5.1: output

```
$$$$
$$$
$$
$
```

Problem 5.2 *Divide I* (1 point)

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

Write a function `void divby5(float arr[], int size)` that divides by 5 all elements of an array. Your program should print in the `main()` function the elements of the array before and after the division. Test your program with an array that contains the following values:
$5.5, 6.5, 7.75, 8.0, 9.6, 10.36$.
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 5.2: input

Testcase 5.2: output

```
Before:
5.500 6.500 7.750 8.000 9.600 10.360
After:
1.100 1.300 1.550 1.600 1.920 2.072
```

Problem 5.3 *Determine lowercase characters* (1 point)

Due by Monday, October 7th, 23:00 **Graded manually**
Language: C

Write a function `int count_lower(char* str)` that counts the number of lowercase characters within in a string. Then write a program where you repeatedly read a string and determine and print the number of lowercase characters in that string. If you provide an empty string (the string will just contain '`\n`'), then the program should stop its execution. You must use a pointer to walk through the string.
You can assume that the string will be not longer than 50 characters and will be valid.

Problem 5.4 *Divide II* (1 point)

Due by Monday, October 7th, 23:00 **Graded manually**

Language: C

Modify your solution for *Divide I* such that you first read an integer `n`, and then elements of an array with `n` components. Therefore you will need to dynamically allocate your array. Then divide by `5` the elements using the `divby5()` function and print the result from the `main()` function. Do not forget to release the allocated memory when not needed anymore.
You can safely assume that the input will be valid.

Problem 5.5 *Computing the scalar product of two vectors* (1 point)

Due by Monday, October 7th, 23:00 **Graded automatically with testcases only**

Language: C

Write a program that reads a number `n`, and then two vectors `v` and `w` of real numbers (of type `double`) with `n` components. Write a function that computes the scalar product of these two vectors. The scalar product is defined as:

$$v \cdot w = \sum_{i=1}^{n} v_i \cdot w_i$$

Use the function to compute the scalar product of the two vectors you read. From the `main()` function print the value of the scalar product on the screen. Additionally write functions for determining and printing on the screen the smallest and largest components of the vector `v`, and the position in the vector where they occur.
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 5.5: input

```
3
1.1
2.5
3.0
1.0
1.0
1.0
```

Testcase 5.5: output

```
Scalar product=6.600000
The smallest = 1.100000
Position of smallest = 0
The largest = 3.000000
Position of largest = 2
The smallest = 1.000000
Position of smallest = 0
The largest = 1.000000
Position of largest = 0
```

Problem 5.6 *Pointer arithmetic* (2 points)

Due by Monday, October 7th, 23:00 **Graded manually**

Language: C

Write a program that counts the number of elements in an array until encountering the first negative value without the usage of any integer counter variables (except for the loop for reading the elements of the array), but with the usage of pointers and pointer arithmetic.
Your program should read an `int` for the length of the array and an array of floats (containing at least one negative value) from the standard input. The number of non-negative values before the first negative value should be printed on the standard output.
You can assume that the input will be valid and the array will always contain at least one negative value. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.6: input

```
5
1.2
3.4
-5.86
4
5.45
```

Testcase 5.6: output

```
Before the first negative value: 2 elements
```

Problem 5.7 *Concatenating two strings* (2 points)

Due by Monday, October 7th, 23:00 **Graded automatically with testcases only**

Language: C

Write a program to concatenate two strings (with the length of at most 100 characters) putting the result in a dynamically allocated array of chars with exact size.
Your program should read from the standard input two strings which may be statically allocated. The dynamically allocated string containing the concatenation result of the two strings should be printed on the standard output.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.7: input

```
one
two
```

Testcase 5.7: output

```
Result of concatenation: onetwo
```

Problem 5.8 *Dynamically allocated matrix multiplication* (3 points)

Due by Monday, October 7th, 23:00 **Graded manually**

Language: C

Write a program that computes the multiplication of two dynamically allocated matrices.
Your program should dynamically allocate the memory for the three matrices (two input matrices and the result matrix). You should write functions for reading a matrix from the standard input, printing a matrix to the standard output, and finally a function for computing the multiplication of two matrices. At the end, do not forget to deallocate the memory used by the three matrices.
The product of two matrices A and B of dimensions $n \times m$ and $m \times p$ can be calculated as:

$$C_{ik} = \sum_{j=1}^{m} A_{ij} \cdot B_{jk}, \text{ with } i = 1, \ldots, n \text{ and } k = 1, \ldots, p.$$

Your program should read three integers (the dimensions `n`, `m` and `p`) and the elements of two integer matrices from the standard input. The result of the matrix multiplication should be printed on the standard output.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.8: input

```
2
2
2
1
2
3
4
1
0
0
1
```

Testcase 5.8: output

```
Matrix A:
1 2
3 4
Matrix B:
1 0
0 1
The multiplication result AxB:
1 2
3 4
```

Bonus Problem 5.9 *Printing dynamically allocated 3D-array sections* (4 points)

Due by Monday, October 7th, 23:00 **Graded automatically with testcases only**

Language: C

Write a program that dynamically allocates memory for a 3D-array of integers and prints the 2D-sections parallel to the "XOY axis" (considering the array dimensions column-dimension, row-dimension and depth-dimension similar to the geometrical X, Y and Z dimensions) of a 3D-array.

Your program should read three integer values corresponding to the dimensions of a 3D-array (in the order of rows, columns, depth) and should dynamically allocate the memory for this 3D-array. You should write functions for reading the elements of the 3D-array from standard input (first iterating through rows, then columns and then the depth) and finally a function for printing the 2D-sections of the 3D-array which are parallel to the "XOY axis". Do not forget about the allocation and deallocation of the memory.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.9: input

```
2
2
3
1
2
3
1
2
3
1
2
3
1
2
3
```

Testcase 5.9: output

```
Section 1:
1 1
1 1
Section 2:
2 2
2 2
Section 3:
3 3
3 3
```

Problem 5.10 *Print numbers counting down* (1 point)

Due by Monday, October 7th, 23:00 **Graded manually**
Language: C
Write a program which reads a positive integer `n` from the keyboard. Then write and call a recursive function for printing the numbers $n, n-1, \ldots, 1$.
You can safely assume that the input will be valid.

Problem 5.11 *Determine if a number prime* (1 point)

Due by Monday, October 7th, 23:00 **Graded automatically with testcases only**
Language: C
Write a program which reads a positive integer `x`. Then write a recursive function for determining if `x` is a prime number or not. The function should return `1` if the number is prime and `0` if not. Print a corresponding message from the `main()` function.
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 5.11: input

```
26
```

Testcase 5.11: output

```
26 is not prime
```

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
   a5_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, October 7th, 23:00.

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.

Assignment 3 - Loops, Arrays, Functions and Strings

- 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 3.1 *Writing numbers* (1 point)

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

Write a program where you first enter a float `x` and then an integer `n` from the keyboard. Print the float `x` `n` times to the screen. Make sure that `n` will have a valid integer value. In the invalid case repeat entering `n` until a valid value will be entered.
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 3.1: input

```
1.25
-8
0
4
```

Testcase 3.1: output

```
Input is invalid, reenter value
Input is invalid, reenter value
1.250000
1.250000
1.250000
1.250000
```

Problem 3.2 *Writing characters* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C

Write a program where you first enter a lowercase character `ch` and then an integer `n` from the keyboard. Print the characters `ch`, `ch−1`, ..., `ch−n` on the screen.
You can safely assume that the input is valid and you do not have to do any checks.

Problem 3.3 *Centimeters to kilometers* (1 point)

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

Write a program that converts an integer length that is entered from the keyboard in cm to km. Write and use a function `float convert(int cm)` that does the actual conversion.
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 3.3: input

```
12
```

Testcase 3.3: output

```
Result of conversion: 0.000120
```

Problem 3.4 *Wrong position* (1 point)

Due by Monday, September 23rd, 23:00 **Graded manually**

Language: C

The program below should print the position of the first occurrence of the character 'g' within a string. You can safely assume that 'g' will be contained in the string. Why does it always print the position 0? Fix the program such that it prints the correct position.

```
#include <stdio.h>

int position(char str[], char c)
{
    int idx;
    /* Loop until end of string */
    for (idx = 0; str[idx] != c && str[idx] != '\0'; ++idx)
        /* do nothing */
    return idx;
}

int main() {
    char line[80];
    while (1) {
        printf("Enter string:\n");
        fgets(line, sizeof(line), stdin);
        printf("The first occurrence of 'g' is: %d\n", position(line, 'g'));
    }
}
```

Problem 3.5 *Computing the average of temperature values* (2 points)

Due by Monday, September 23rd, 23:00 **Graded manually**

Language: C

Write a program where you first enter a character `c` followed by an integer `n` and `n` double values representing temperatures in Celsius. Use an array for storing the temperatures. You can assume that not more than 100 temperature values would be introduced. Your program should compute and/or print the following: if `c` is 's' the sum of the temperatures, if `c` is 'p' the list of all temperatures, if `c` it 't' the list of all temperatures in Fahrenheit and if another character was introduced then the arithmetic mean (or average) of all temperatures. The formula for converting Celsius to Fahrenheit is the following:

$$F = \frac{9}{5} \cdot C + 32.$$

Use a `switch` instruction in your solution.
You can safely assume that the input will be valid.

Problem 3.6 *Kilograms and grams to pounds* (1 point)

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

Language: C

Write a program that converts the units of mass kg and g to pounds. First read the weight of an object expressed by values for kilograms and grams from the keyboard and convert the units of mass using the function (written by you as well)

```
float to_pounds(int kg, int g);
```

that does the actual conversion. Note that:
1 kilogram = 2.2 pounds
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 3.6: input

```
5
100
```

Testcase 3.6: output

```
Result of conversion: 11.220000
```

Problem 3.7 *Printing a form* (1 point)

Due by Monday, September 23rd, 23:00 **Graded manually**

Language: C

Write a program which reads two integers `n`, `m` and a character `c` from the keyboard. This program should define and call a function with the prototype:

```
void print_form(int n, int m, char c);
```

which prints a trapezoid of height `n`, bases `m` and `m+n-1` consisting of the character `c` as in the following testcase.

Testcase 4.4: input

```
4
3
$
```

Testcase 4.4: output

```
$$$
$$$$
$$$$$
$$$$$$
```

You can safely assume that the input will be valid.

Problem 3.8 *Computing sum and average* (1 point)

Due by Monday, September 23rd, 23:00 **Graded manually**

Language: C

Write a program where you can enter from the keyboard up to 10 floats. If the number entered is equal to -99.0, stop reading numbers from the keyboard and compute the sum and average of all values (excluding -99.0) using two functions (one for the sum and one for the average). Print your results on the screen.

You can safely assume that the input will be valid.

Make sure you consider all the cases: less than 10 numbers might be entered. After all the numbers have been entered you need to make sure that the sum and average are computed.

Bonus Problem 3.9 *Determine sum of two values in array* (1 point)

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

Language: C

Write a program which reads from the keyboard an integer value `n` followed by `n` double values as elements of an array with not more than 20 elements. Write also a function with the prototype:

```
double sum25(double v[], int n);
```

which computes and returns the sum of the elements in `v` at position 2 and position 5. Check that positions 2 and 5 are valid within `v`, if not then print a corresponding message on the screen. In this case the function should return the value -111. Print the elements of the array and the sum on the screen.

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 3.9: input

```
6
1.5
1.5
1.5
1.5
1.5
-1.5
```

Testcase 3.9: output

```
sum=0.000000
```

Problem 3.10 *Return changes by reference* (1 point)

Due by Monday, September 23rd, 23:00 **Graded manually**
Language: C

Write a program which reads two float values from the keyboard. Then write three functions. The first function should return the product of the two float values and should have the prototype:

```
float product(float a, float b);
```

The second function should return the product of the two float values by reference and should have the prototype:

```
void productbyref(float a, float b, float *p);
```

The third function should add 3 to the first float and 11 to the second float and return the change by reference. It should have the prototype:

```
void modifybyref(float *a, float *b);
```

Show that the calls of the first two functions have the same effect. Also show what is the effect of calling `modifybyref`.
You can safely assume that the input will be valid.

Problem 3.11 *Working with strings* (2 points)

Due by Monday, September 23rd, 23:00 **Graded automatically with testcases only**
Language: C

Write a program where you can enter two strings called `one` and `two` from the keyboard. The string should be able to contain spaces. The program should do the following:

1. Print on the screen the lengths of both strings,
2. Print on the screen the concatenation of `one` with `two`,
3. Declare a third string, copy correctly `two` into it and print the third string to the screen,
4. Compare the two strings `two` and `one` and print a corresponding message to the screen,
5. Read a character `c` from the keyboard and search for `c` in `two`. The position of the first occurrence of `c` within `two` should be printed to the screen. If the character is not contained in the string then print a corresponding message on the screen.

For solving this problem use the string functions from `string.h`.
Learn how to use them with the help of the man pages.
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 3.11: input

```
first string
hello world
l
```

Testcase 3.11: output

```
length1=12
length2=11
concatenation=first stringhello world
copy=hello world
one is smaller than two
position=2
```

Bonus Problem 3.12 *Detect all occurrences of a character* (1 point)

Due by Monday, September 23rd, 23:00 **Graded automatically with testcases only**
Language: C

Modify your solution for **Problem 3.11** such that all occurrences of the character `c` can be detected in `two`. All positions of occurrence should be printed to the screen.
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 3.12: input

```
first string
hello world
l
```

Testcase 3.12: output

```
length1=12
length2=11
concatenation=first stringhello world
copy=hello world
one is smaller than two
position=2
position=3
position=9
```

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
   a3_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 23rd, 23:00.

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

Course: CH-230-A
September 10th, 2019

Assignment 2 - Reading from the Keyboard, Using Pointers, Conditions and Loops

- 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 2.1 *Reading from the keyboard* (1 point)

Presence assignment, due by 11:00 AM today **Graded automatically with testcases only**
Language: C
Write a program which does the following:

1. reads two doubles from the keyboard,
2. prints the sum of the two doubles,
3. prints the difference of the two doubles (first minus second),
4. prints the square of the first double,
5. reads two integers from the keyboard,
6. computes the sum and product of the two integers,
7. prints the sum and product of the integers,
8. reads two chars from the keyboard,
9. computes the sum and product of the two chars,
10. prints the sum and product of the chars as decimal values and as chars.

You can assume that the input will be correct.
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 2.1: input

```
1.2
3.6
2
5
(
)
```

Testcase 2.1: output

```
sum of doubles=4.800000
difference of doubles=-2.400000
square=1.440000
sum of integers=7
product of integers=10
sum of chars=81
product of chars=1640
sum of chars=Q
product of chars=h
```

Problem 2.2 *Decimal, octal and hexadecimal numbers* (1 point)

Presence assignment, due by 11:00 AM today **Graded automatically with testcases only**
Language: C
Write a program which does the following:

1. reads a char from the keyboard,
2. and prints the char as character as well as in decimal, octal and hexadecimal notation.

You can assume that the input will be correct.
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 2.2: input

```
{
```

Testcase 2.2: output

```
character={
decimal=123
octal=173
hexadecimal=7b
```

Problem 2.3 *Time calculation* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C
Write a program where you can enter integer numbers for weeks, days and hours as input from the keyboard. Your program should compute and output by printing on the screen the total number of hours.
You can assume that the input will be correct.

Problem 2.4 *Area computations* (1 point)

Due by Monday, September 16th, 23:00 **Graded automatically with testcases only**
Language: C
Write a program that reads from the keyboard three float values for the variables `a`, `b` and `h`. Compute and print on the screen the areas of: the square with the side `a`, the rectangle with the length `a` and the width `b`, a triangle with the base `a` and the height `h`, and a trapezoid with the bases `a`, `b` and the height `h`.
You can assume that the input will be correct.
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 2.4: input

```
10
14.5
5
```

Testcase 2.4: output

```
square area=100.000000
rectangle area=145.000000
triangle area=25.000000
trapezoid area=61.250000
```

Problem 2.5 *Pointers and their content* (1 point)

Due by Monday, September 16th, 23:00 **Graded manually**
Language: C
Write a program which reads an integer variable `a` from the keyboard and prints the value on the screen. Then declare and initialize a pointer `ptr_a` pointing to `a`, print the address contained in the pointer variable on the screen, increase the value of `a` by `5` by using the pointer variable and print the modified value and the address of the variable on the screen as well.
You can safely assume that the input will be correct.

Problem 2.6 *Multiple pointers to same data* (1 point)

Due by Monday, September 16th, 23:00 **Graded manually**
Language: C
Write a program which reads two double variables `x` and `y` from the keyboard. Then declare and initilize three pointers `ptr_one`, `ptr_two` and `ptr_three` such that `ptr_one` and `ptr_two` will both point to the variable `x` and `ptr_three` will point to `y`. By using `printf` show that `ptr_one` and `ptr_two` contain the same memory address and `ptr_three` contains a different address.
You can assume that the input will be correct.

Problem 2.7 *Infinite loop by bad coding* (1 point)

Due by Monday, September 16th, 23:00 **Graded manually**

Language: C

The program below prints

```
i is 8
i is 8
...
```

until you stop the execution by pressing `Ctrl-C`. Fix the program such that it prints 8, 7, 6, 5 and 4 as values for `i`.

```
#include <stdio.h>
int main()
{
    int i = 8;
    while (i >= 4)
        printf("i is %d\n", i);
        i--;
    printf("That's it.\n");
    return 0;
}
```

Problem 2.8 *Divisible by 2 and 7?* (1 point)

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

Language: C

Write a program, where you can enter an integer from the keyboard. Determine whether the number is divisible by both 2 and 7. Then either print on the screen
"`The number is divisible by 2 and 7`" or
"`The number is NOT divisible by 2 and 7`".
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 2.8: input

```
56
```

Testcase 2.8: output

```
The number is divisible by 2 and 7
```

Problem 2.9 *Categorization of characters* (1 point)

Due by Monday, September 16th, 23:00 **Graded automatically with testcases only**

Language: C

Write a program where you can enter a character from the keyboard. Then determine whether the character is a digit or a letter or some other symbol and print a corresponding message 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 2.9: input

```
!
```

Testcase 2.9: output

```
! is some other symbol
```

Problem 2.10 *Days and hours* (2 points)

Due by Monday, September 16th, 23:00 **Graded manually**

Language: C

Write a program where you can enter an integer `n` from the keyboard. Then a conversion table for 1 to `n` days should be printed on the screen as in the example below. Make sure that the integer value you entered for `n` is valid (positive and non-zero). If an invalid integer `n` was entered then repeat the entering until a valid value will be entered.
Use a `while` loop in your solution.

```
1 day = 24 hours
2 days = 48 hours
3 days = 72 hours
...
```

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
   a2_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 16th, 23:00.

Assignment 1 - Fixing and Writing Simple Programs

- 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 1.1 *Compute division* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C

Fix the program below such that it prints the correct result. Why is the result 0.000? Write your answer and explanations within comments.

```
#include <stdio.h>

int main() {
    double result; /* The result of our calculation */
    result = (3 + 1) / 5;
    printf("The value of 4/5 is %lf\n", result);
    return 0;
}
```

Problem 1.2 *Wrong output* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C

Fix the program below such that it prints the correct value. Why does the program print "`The result is -1073745604`"? (Values will vary). Write your answer and explanations within comments.

```
#include <stdio.h>

int main() {
    int result; /* The result of our calculation */
    result = (2 + 7) * 9 / 3;
    printf("The result is %d\n");
    return 0;
}
```

Problem 1.3 *A compile error* (1 point)

Due by Monday, September 9th, 23:00 **Graded manually**
Language: C

You will get compiler errors, when you try to compile the example code given below.
Read the error messages that the compiler produces and fix the errors such that your source code compiles successfully. Then fix the program to print the correct result. Explain within comments the reason of the errors and describe your fixes.

```
include <stdio.h>

int main() {
    float result; /* The result of the division */
    int a = 5;
    int b = 13.5;
    result = a / b;
    printf("The result is %d\n", result);
    return 0;
}
```

Problem 1.4 *Simple arithmetics* (1 point)

Due by Monday, September 9th, 23:00 **Graded automatically with testcases only**

Language: C

Write a program which does the following:

1. assigns 17 to `x` and 4 to `y`,
2. prints the values of `x` and `y`,
3. computes the sum of `x` and `y` and prints the result,
4. computes the product of `x` and `y` and prints the result,
5. computes the difference of `x` and `y` (`x` minus `y`) and prints the result,
6. computes the division of `x` and `y` (`x` divided by `y`) and prints the result (the result should be a `float`),
7. computes the remainder of the division of `x` and `y` in this order and prints the result.

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 1.4: input

Testcase 1.4: output

```
x=17
y=4
sum=21
product=68
difference=13
division=4.250000
remainder of division=1
```

Problem 1.5 *Using `printf` for multiple data types and conversions* (1 point)

Due by Monday, September 9th, 23:00 **Graded automatically with testcases only**

Language: C

Write a program which:

1. declares and initializes an integer variable `x` with 2138, and prints the value of `x` over 9 places,
2. declares and initializes a float variable `y` with -52.358925, and prints the value of `y` over 11 places and with a floating point precision of 5,
3. declares and initializes a char variable `z` with 'G', and prints the character on the screen,
4. declares and initializes a double variable `u` with 61.295339687, and prints the value of `u` with a floating point precision of 7.

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 1.5: input

Testcase 1.5: output

```
x=     2138
y=  -52.35892
z=G
u=61.2953397
```

Problem 1.6 *Printing a char as character and as decimal value* (1 point)

Due by Monday, September 9th, 23:00 **Graded manually**

Language: C

Write a program which declares and initializes a char variable `c` with 'F' and prints on the screen the third character (within the alphabet) after `c` as a character and as the corresponding ASCII code using only arithmetic operations.

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
   a1_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 9th, 23:00.