J.NAWROCKI, M. ANTCZAK, H. ĆWIEK, W. FROHMBERG, A. HOFFA, M. KIERZYNKA, S. WĄSIK

IMPERATIVE PROGRAMMING

C LANGUAGE

EX. 1. Write in C language programs described below, compile them and tentatively debug:

a) Program that prints to the screen *„Actium Battle”*. Try to modify the program source code in order to learn different warnings and errors that the compiler will generate when you add additional or remove required characters such as quotation mark (‘`"`’), semicolon (‘`;`’), brackets (‘`(`’, ‘`)`’) or new line (‘`\n`’). The example of the correct program source code was defined in “*Imperative programming. C language*” lecture in the slide 9.

b) Program that computes and prints to the screen the sum of two integer numbers 18 and 2.

c) Program that computes and prints to the screen the sum of any two input integer numbers.

d) Program that computes and prints to the screen the minimal value of any two input integer numbers.

e) Program that computes and prints to the screen the absolute value of any input integer number.

f) Program that computes and prints to the screen the number of digits of any natural number *(0, 1, 2, ...)*.

The last program should be written using *“incremental”* method of software implementation. It means that you have to start from writing, compiling and running a stub of the final source code which was described in the slide 145. Next you should extend your implementation by adding additional code fragments that can be found in slides 148, 150 and 152.

EX. 2. Symbolic execution of an algorithm which is described by block diagram is achieved by systematic *„walking”* through block diagram paths and observing how instructions that are executed change values of variables (look at lecture slides from 105 to 142). The symbolic execution of the algorithm which is implemented in C language involves:

1. *„Source code instrumentalization”* – inserting additional *printf(...)* instructions between original source code instructions which can be used to observe how the values of selected variables are modified during the program execution.

2. Execution of the program after adding instrumentalization.

It is a very simple form of *debugging*.

Let us assume that we have the program which is implemented using source code presented below:

Line no.	C language instruction
1	`int main() {`

(*) asterisk marks problems which are not solved during exercise classes and should be solved as a homework

2	`int   X;`
3	`scanf("%d", &X);`
4	`if ( X < 0 ) {`
5	`   X = -X;`
6	`   }`
7	`printf("%d \n", X);`
8	`return;`
9	`}`

The source code of the above program after instrumentalization process used to observe modification of the variable X value is presented below (instrumentalization instructions are printed in red):

```
int main() {
         int   X;
         printf("--- 2---: X=%d \n", X);
         scanf("%d", &X);
         printf("--- 3---: X=%d \n", X);
         if ( X < 0 ) {
              X = -X;
              printf("--- 5---: X=%d \n", X);
         }
         printf("%d \n", X);
         return;
}
```

Write the source code of the program after instrumentalization process that can be used to solve the problem from point 1F. Then verify the correct/incorrect behavior of your solution with execution of the instrumentalized program.

EX. 3. **An automorphic number** is a number whose square ends with the same digits as the number itself. For example, 5^2 = 2**5**, 6^2 = 3**6** and **25**^{2} = 6**25** so 5, 6 and 25 are all automorphic numbers. Write the program which can be used to answer the question if the natural number *N* defined by the user is the automorphic number. During implementation you should use *"incremental"* method of the software implementation (look at hints defined at the end of task 1). Next observe how variables defined in your solution source code are changed during the program execution after instrumentalization process.

Hint 1. In the C language the value of expression ***a/b*** where a and b are integer numbers is an integer number obtained from division, moreover the value of expression ***a%b*** is a remainder.

Hint 2. While solving this problem a good idea is to design and implement additional helper function which can be called *Order*. This function computes the relation defined as following:

Order (n) is a minimal integer number 10^k which satisfies the equation $n < 10^k$.

For example *Order (1) = Order (9) = 10, Order (10) = Order (99) = 100* and so on.

ZAD. 4. Write the program which can be used to find all automorphic numbers in the range from *A* to *B* which will be defined by user. You should use the function *Automorf(n)* which was implemented in the solution of task 3.

(*) asterisk marks problems which are not solved during exercise classes and should be solved as a homework

In all homework exercises defined below:

1. **Design test cases which will help to design user friendly interface.**
2. **Draw a block diagram of the solution designed by you.**
3. **Analyze (using manual simulation method which was presented during the lecture) selected test cases in order to confirm the correctness of proposed solution which was designed using the block diagram.**
4. **Based on the proposed solution write the program in the C language.**
5. **Confirm correctness of the proposed implementation through test sets execution.**

EX. 5(*). Write a program that computes and prints to the screen the value of an arithmetic mean of three integer numbers.

EX. 6(*). Write a program that takes three integer numbers as an input and prints the answer to the question if these values can be **lengths of sides of some triangle**. **Hint:** *|b-c|<a<b+c*.

EX. 7(*).Write a program that computes and prints to the screen the value of a **factorial *n!*** where *n* is a natural number (for example 0, 1, 2, .. 8).

EX. 8(*). Write a program that computes and prints to the screen the value of **the sum of digits** of some natural number (for example the sum of digits of 0 is equal to 0, but the sum of digits of 99 is equal to 18).

EX. 9(*). Write a program that computes and prints to the screen the value of **exponentiation function a^b** where *a* and *b* are natural numbers (0,1, 2,..).

EX. 10(*). Write a program that reads a finite sequence of integer numbers and then prints them to the screen in a reversed order (for example sequence of 10 numbers).

EX. 11(*). Write a program which can be used to find **the minimal number** of the finite sequence of natural numbers defined by the user (for example sequence of 10 numbers).

EX. 12(*). Write a program that computes and prints to the screen **the sum** of the finite sequence of integer numbers defined by the user. Then modify it to use floating point numbers (for example sequence of 10 numbers).

EX. 13(*). Write a program that computes and prints to the screen **the arithmetic mean** of the finite sequence of integer numbers defined by the user. Then modify it to use floating point numbers (for example sequence of 10 numbers). To solve this problem you should modify the solution that was proposed for the exercise 12.

EX. 14(*). Write a program that reads the size of a matrix (one natural number) and as a result draws to the screen the *left-bottom matrix* of size *n* x *n*. In the left-bottom matrix on a main diagonal and below it the character '*X*' is printed and above the main diagonal the space is printed. The left-bottom matrix example for size *n=4* is presented below:

```
X
XX
XXX
XXXX
```

(*) asterisk marks problems which are not solved during exercise classes and should be solved as a homework

EX. 15(*). Write a program that reads the size of a matrix (one natural number) and as a result draws to the screen the *right-top matrix* of size *n* x *n*. In the right-top matrix on a main diagonal and above it the character '*X*' is printed and below the main diagonal the space is printed. The right-top matrix example for size *n=4* is presented below:

```
XXXX
 XXX
  XX
   X
```

EX. 16(*). Write a program that computes all possible solutions of a **quadratic equation** $ax^2 + bx + c = 0$. You have to take into consideration all possible cases of coefficients values (a, b, c).

EX. 17(*). **A prime number** is a natural number that has exactly two *distinct* natural divisors: 1 and itself. Write the program which can be used to answer the question if the natural number *N* defined by user is the prime number. **Hint:** You can use *sieve of Eratosthenes*.

EX. 18(*). Write a program that finds and prints to the screen **all prime numbers from some range** which is defined by the user. **Hint:** You can use solution from above exercise.

EX. 19(*). Write a program that computes and prints to the screen the value of a **greatest common divisor** (GCD) of any two natural numbers given as an input.

EX. 20(*). Write a program that computes and prints to the screen the value of a **greatest common divisor** (GCD) for finite set of natural numbers defined by the user (for example sequence of 10 numbers).

EX. 21(*). Write a program that computes and prints to the screen the value of a **least common multiple** (LCM) of any two natural numbers given as an input.

EX. 22(*). Write a program that computes and prints to the screen the value of a **least common multiple** (LCM) for finite set of natural numbers defined by the user (for example sequence of 10 numbers).

EX. 23(*). Write a program that searches the defined by user range of numbers *<D, G>* in order to find and print to the screen all possible pairs of natural numbers *a, b* for which exists natural number *c* such that the equation $a^2 + b^2 = c^2$ is satisfied (for example $3^2 + 4^2 = 5^2$).

EX. 24(*). Write a program that finds and prints to the screen the value of ***n-th* Fibonacci number**, where *n* is a natural number.

EX. 25(*). Write a program that sorts the finite sequence of integer numbers defined by the user using the **selection-sort algorithm** and prints the sorted sequence to the screen.

EX. 26(*). Write a program that **merges several (at least two) finite, sorted sequences of integer numbers** having not necessarily equal lengths into the one sorted sequence. The result should be printed to the screen.

EX. 27(*). Write a program that computes the **quadratic matrices multiplication** for any two quadratic matrices given as an input.

EX. 28(*). Write a program which can be used to check the **collinearity of three points on the plane**. Every point is represented by two coordinates *(X,Y)* defined by the user.

NON-OBLIGATORY EXCERCICES FOR THE MOST EXPERIENCED STUDENTS

(*) asterisk marks problems which are not solved during exercise classes and should be solved as a homework

EX. 29. Programs that prints to the screen the exact copy of their source code are called **“quine programs”** from the name of the logician Willard van Orman Quine. Analyse following programs and try to understand them. Can you write your own, different quine?

Hint: In the program below the second and the third lines should be written in the same line!

```
#include<stdio.h>
char *program="#include<stdio.h>%cchar *program=%c%s%c;%cvoid
main()%c{%cprintf(program,10,34,program,34,10, 10,10,10);%c}";

void main()
{
printf(program,10,34,program,34,10,10,10,10);
}
```

And here a bit shorter version:

```
p="p=%c%s%c;main(){printf(p,34,p,34);}";main(){printf(p,34,p,34);}
```

EX. 30. Write **as short as you can** program which merges three sorted sequences having length **n** into the one sorted sequence having length **3*n**. The program should read from the user number **n** and then three sequences of **n** numbers. The result should be printed to the screen. The program should work in linear time that means that it should scan each sequence only once. The length of the code should be determined based on **the number of bytes** used to save the program’s source file (the one that can be found in file properties).

Example input:

```
4
1 4 7 8
3 3 5 10
2 7 9 11
```

Expected output:

```
1 2 3 3 4 5 7 7 8 9 10 11
```

Hint: During the compilation your program can generate some warnings. It is better to use C than C++ because it is much less strict language.

(*) asterisk marks problems which are not solved during exercise classes and should be solved as a homework