Computer Science

Laboratory Exercise 4

Objectives

- Repeated execution of instructions to reach a certain goal
- Data input processing of many values

Technical contents

- Use of for and while for loop execution
- Understand how nested loops work
- String manipulation

To solve during the lab

1. Write programs that read a sequence of integer inputs and print
    a. The smallest and largest of the inputs.
    b. The number of even and odd inputs.
    c. Cumulative totals. For example, if the input is 1 7 2 9, the program should print 1 8 10 19.
    d. All adjacent duplicates. For example, if the input is 1 3 3 4 5 5 6 6 6 2, the program should print 3 5 6.

2. Write programs that read a line of input as a string and print
    a. Only the uppercase letters in the string.
    b. Every second letter of the string.
    c. The string, with all vowels replaced by an underscore.
    d. The number of digits in the string.
    e. The positions of all vowels in the string.

3. Write a program that reads an integer and displays, using asterisks, a square and a filled diamond of the given side length. For example, if the side length is 4, the program should display:

```
****
****
****
****
   *
  ***
 *****
```

```
*******
 *****
  ***
   *
```

4. Write a program that reads a word and prints the word in reverse. For example, if the user provides the input *"Harry"*, the program prints:
   `yrraH`

To solve at home

5. *Prime numbers*. Write a program that prompts the user for an integer and then prints out all prime numbers up to that integer. For example, when the user enters 20, the program should print
   ```
   2357
   11
   13
   17
   19
   ```
   Recall that a number is a prime number if it is not divisible by any number except 1 and itself.

6. Write a program that reads a word and prints all substrings, sorted by length. For example, if the user provides the input "rum", the program prints:
   ```
   r
   u
   m
   ru
   um
   rum
   ```

7. *The game of Nim*. This is a well-known game with a number of variants. The following variant has an interesting winning strategy. Two players alternately take marbles from a pile. In each move, a player chooses how many marbles to take. The player must take at least one but at most half of the marbles. Then the other player takes a turn. The player who takes the last marble loses. Write a program in which the computer plays against a human opponent. Generate a random integer between 10 and 100 to denote the initial size of the pile. Generate a random integer between 0 and 1 to decide whether the computer or the human takes the first turn. Generate a random integer between 0 and 1 to decide whether the computer plays *smart* or *stupid*. In stupid mode the computer simply takes a random legal value (between 1 and $n/2$) from the pile whenever it has a turn. In smart mode the computer takes off enough marbles to make the size of the pile a power of two minus 1—that is, 3, 7, 15, 31, or 63. That is always a legal move, except when the size of the pile is currently one less than a power of two. In that case, the computer makes a random legal move. You will note that the computer cannot be beaten in smart mode when it has

the first move, unless the pile size happens to be 15, 31, or 63. Of course, a human player who has the first turn and knows the winning strategy can win against the computer.