Introduction to programming

Labor09

Revision

- Sets
  - empty set: set()
  - functions
  - Union, intersection, difference, symmetric difference
- Dictionaries
  - empty dictionary: dict(), {}
  - functions
  - access dictionary items
- Lambda functions

Homework 1 – Club loyalty

- https://progcont.hu/progcont/100360/?pid=201553

Homework 1 – Club loyalty

- Some athletes are loyal to a single club throughout their entire sporting careers. Others are not like that: they change clubs from time to time to advance their careers or ensure their financial well-being.
- Write a program that reads an integer from the standard input in the first line, and then reads the data of sports clubs from the subsequent lines, until the end-of-file (EOF). Each line contains the names of athletes who have ever been associated with the respective sports club in the following format:
- `club_name:athlete_name[,athlete_name]...`
- The program should write to the standard output the names of athletes who have been associated with more clubs than the specified integer! If there are multiple such athletes, order them based on the number of clubs they have been part of.
- The more clubs someone has been associated with, the higher they should appear on the list! If it is still impossible to decide between two or more athletes based on this criterion, write their names in lexicographically increasing order, as shown in the sample output.

Homework 1 – Club loyalty

- Sample Input

  -
    ```
    2
    ```

  -
    ```
    Golden Team:Grosics,Buzanszky,Lorant,Lantos,Bozsik,Zakarias,Budai,Kocsis,Hidegkuti,Puskas,Czibor
    ```

  -
    ```
    Mighty Magyars:Buzanszky,Lantos,Zakarias,Hidegkuti
    ```

  -
    ```
    Marvellous Magyars:Grosics,Buzanszky,Bozsik,Hidegkuti,Puskas
    ```

  -
    ```
    Magnificent Magyars:Lorant,Budai,Kocsis,Hidegkuti,Puskas
    ```

  -
    ```
    Magical Magyars:Buzanszky,Lantos,Hidegkuti,Puskas
    ```

- Output for Sample Input

  -
    ```
    Hidegkuti: 5
    ```

  -
    ```
    Buzanszky: 4
    ```

  -
    ```
    Puskas: 4
    ```

  -
    ```
    Lantos: 3
    ```

```python
import sys
n = int(input())
footballers = {}
for line in sys.stdin:
    data = line.split(":")
    names = data[1].strip().split(",")
    for i in range(len(names)):
        if names[i] in footballers:
            footballers[names[i]] += 1
        else:
            footballers[names[i]] = 1

for key, value in sorted(footballers.items(),
            key = lambda x: (-x[1], x[0])):
    if value > n:
        print(f"{key}: {value}")
```

Homework 2
Every day an apple keeps the doctor away

- https://progcont.hu/progcont/100358/?pid=201544

Homework 2
Every day an apple keeps the doctor away

- We have heard the phrase in the title many times: fruit is an essential part of a healthy diet.
- Write a program that reads, line by line from the standard input, the names of fruits and the names of children who like that fruit. The structure of the lines is as follows:

- `fruit_name:child_name[,child_name]...`

- The names of fruits and children are unique; there are no two children with the same name who like the same fruit.
- Your program should write to the standard output the names of children in lexicographically increasing order, along with the number of different fruits each child likes, in the format shown in the sample output.

Homework 2
Every day an apple keeps the doctor away

Sample Input

```
apple:Pete,Kate
plum:John
banana:Kate,John,Charles
pineapple:Hannah,Anna,Penny
apricot:Fanny,Pete
```

Output for Sample Input

```
Anna: 1
Charles: 1
Fanny: 1
Hannah: 1
John: 2
Kate: 2
Penny: 1
Pete: 2
```

Homework 2

Every day an apple keeps the doctor away

```
import sys

children = {}
for line in sys.stdin:
    data = line.split(":")
    names = data[1].strip().split(",")
    for i in range(len(names)):
        if names[i] in children:
            children[names[i]] += 1
        else:
            children[names[i]] = 1

for key, value in sorted(children.items()):
    print(f"{key}: {value}")
```

Homework 3
New curriculum model

- https://progcont.hu/progcont/100278/?pid=201288

Homework 3
New curriculum model

- Write a program that reads an integer (N) from the standard input in the first line. In the following N lines of input, there are subject codes, names, and credit values, one per line, in the following format:

- `subject_code:subject_name:credit_number`

- The subject_code and subject_name are strings, and credit_number is a positive integer.
- The subject_code is always in the format "INBMM-xx-yy," where xx represents the recommended semester number, and yy is the unique identifier for the subject.
- The program should write to the standard output the names of the subjects sorted in ascending order based on the recommended semesters. If multiple subjects have the same recommended semester, list them in increasing order of their unique identifiers on the output!

Homework 3
New curriculum model

- Sample Input:

```
5
INBMM-02-07:Data Structures and Algorithms:6
INBMM-01-01:Algorithms and Fundamentals of Programming:2
INBMM-01-02:Electronics:6
INBMM-03-14:Fundamentals of Economics:6
INBMM-02-08:Mathematics for Engineers 2:6
```

- Output for Sample Input:

```
Algorithms and Fundamentals of Programming
Electronics
Data Structures and Algorithms
Mathematics for Engineers 2
Fundamentals of Economics
```

Homework 3
New curriculum model

```python
n = int(input())
subjects = {}

for i in range(n):
    data = input().split(":")
    subjects[data[0]] = data[1]

for key, value in sorted(subjects.items()):
    print(value)
```

Command line arguments

- It is possible to pass some values from the command line to your Python program when they are executed. These values are called Command Line Arguments.
- So, the arguments that are given after the name of the program in the command line shell of the operating system are known as Command Line Arguments.
- Python provides various ways of dealing with these types of arguments.
- The three most common are:
  - Using sys.argv
  - Using getopt module
  - Using argparse module

Command line arguments

```
argc - number of command line arguments -> int

sys.argv[0] - program name/file location -> str

sys.argv[1] - first command line argument -> str

sys.argv[2] - second command line argument -> str

...

sys.argv[n] - nth command line argument -> str
```

Command line arguments

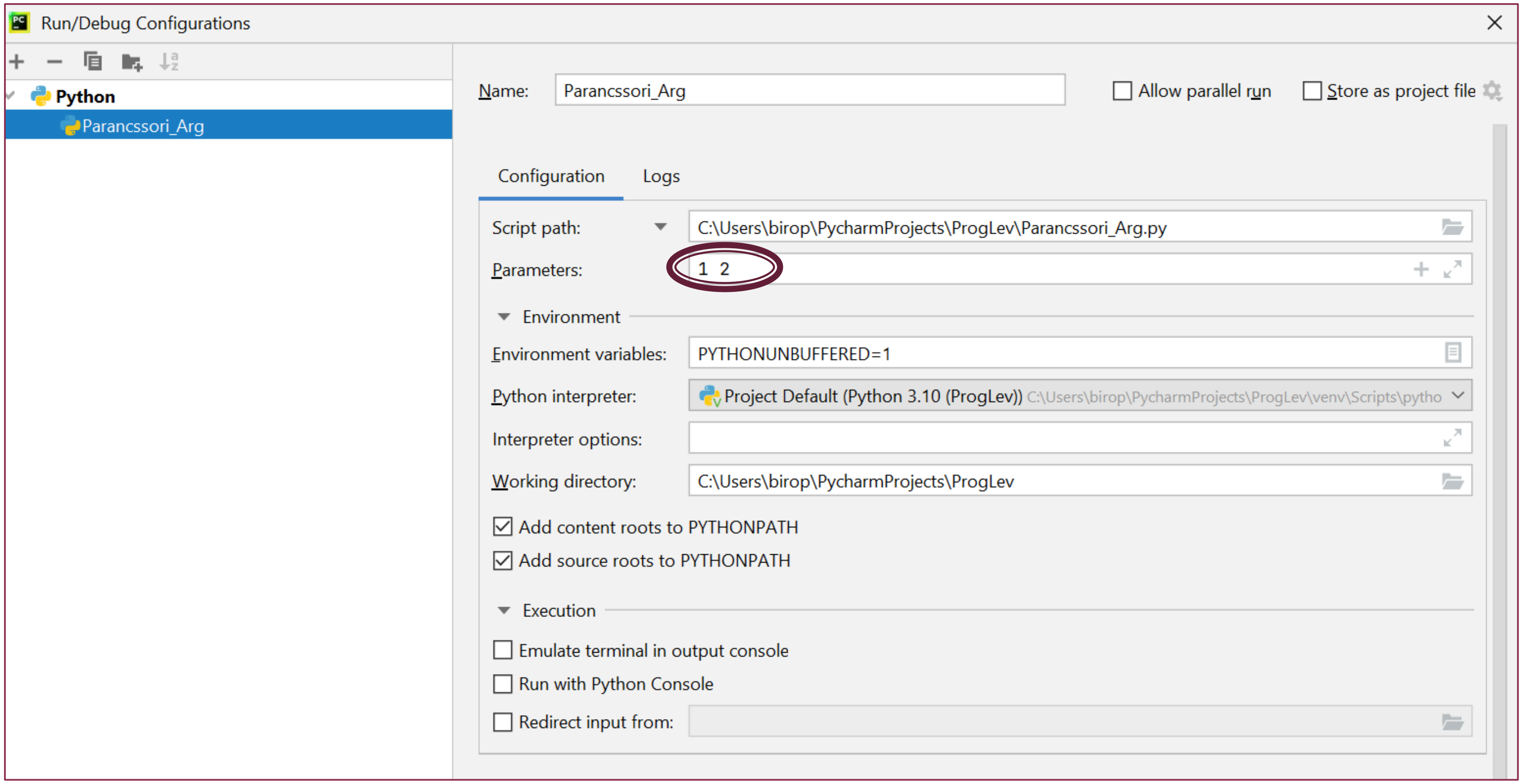
```
import sys

argc = len(sys.argv)
print("Number of arguments:", argc)

print("File location:", sys.argv[0])

print("Arguments:")
for i in range(1, argc):
    print(sys.argv[i])
```

Command line arguments


Run/Debug Configurations

Python
Parancssori_Arg

Name: Parancssori_Arg
☐ Allow parallel run
☐ Store as project file

Configuration | Logs

Script path: C:\Users\birop\PycharmProjects\ProgLev\Parancssori_Arg.py
Parameters: 1 2

Environment
Environment variables: PYTHONUNBUFFERED=1
Python interpreter: Project Default (Python 3.10 (ProgLev)) C:\Users\birop\PycharmProjects\ProgLev\venv\Scripts\pytho
Interpreter options:
Working directory: C:\Users\birop\PycharmProjects\ProgLev
☑ Add content roots to PYTHONPATH
☑ Add source roots to PYTHONPATH

Execution
☐ Emulate terminal in output console
☐ Run with Python Console
☐ Redirect input from:

Command line arguments

RUN – Total Commander -> CMD:

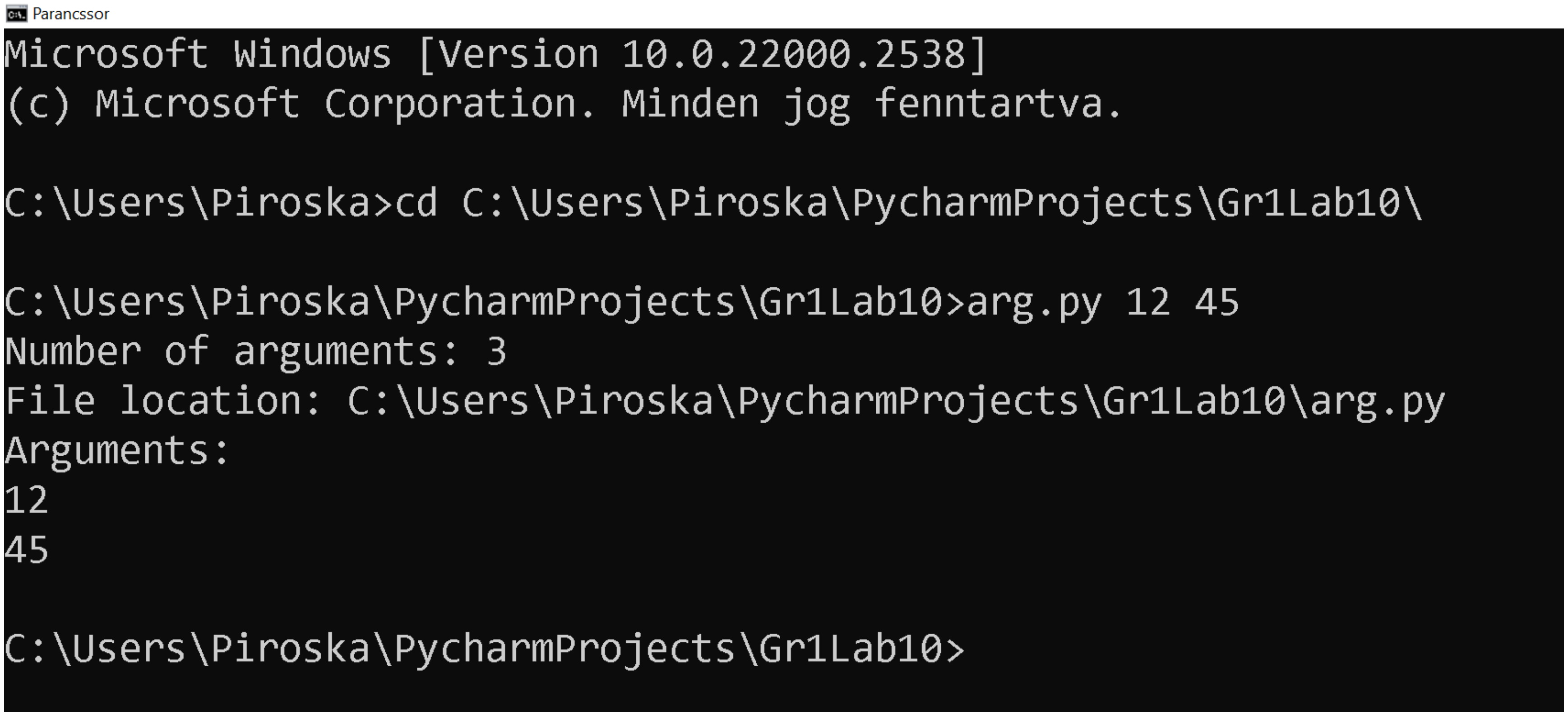
- c:/Users/birop/PycharmProjects/Gr8Lab9/arg.py 1 2

Output:

- Number of arguments: 3
- File locations: c:/Users/birop/PycharmProjects/Gr8Lab9/arg.py
- Arguments:
- 1
- 2

Command line arguments

- CMD – Command Prompt



```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. Minden jog fenntartva.

C:\Users\Piroska>cd C:\Users\Piroska\PycharmProjects\Gr1Lab10\

C:\Users\Piroska\PycharmProjects\Gr1Lab10>arg.py 12 45
Number of arguments: 3
File location: C:\Users\Piroska\PycharmProjects\Gr1Lab10\arg.py
Arguments:
12
45

C:\Users\Piroska\PycharmProjects\Gr1Lab10>
```

Command line arguments

- Determine the product of the command line arguments.

Command line arguments

```python
import sys

argc = len(sys.argv)
print("Number of arguments:", argc)

p = 1
for i in range(1, argc):
    p = p * int(sys.argv[i])
print("Product = ", p)
```

Number of odd elements

The function

- Write a function named **count_of_odds** that returns the number of odd numbers in the command line arguments.

Returned value

- the number of elements matching the condition

https://viskillz.inf.unideb.hu/prog/#/?exercise=P108101c&page=sheet&week=P1081

Number of odd elements

```python
import sys

def count_of_odds() -> int:
    count = 0
    for number in sys.argv[1:]:
        if int(number) % 2 == 1:
            count += 1
    return count

def main():
    print(count_of_odds())

if __name__ == "__main__":
    main()
```

Number of odd elements

```
import sys

def count_of_odds(numbers: list[int]) -> int:
    count = 0
    for number in numbers:
        if number % 2 == 1:
            count += 1
    return count

def main():
    numbers = []
    argc = len(sys.argv)
    for i in range(1, argc):
        numbers.append(int(sys.argv[i]))

    print(count_of_odds(numbers))

if __name__ == "__main__":
    main()
```

Number of prime elements

The function

- Write a function named count_of_primes that returns the number of primes among the command line arguments.

Returned value

- number of elements matching the condition

https://viskillz.inf.unideb.hu/prog/#/?exercise=P108103c&page=sheet&week=P1081

Number of prime elements

```
import sys
import math

def is_prime(n: int) -> bool:
    if n == 2:
        return True
    if n < 2 or n % 2 == 0:
        return False
    for i in range(3,
        int(math.sqrt(n)), 2):
        if n % i == 0:
            return False
    return True
```

```
def count_of_primes() -> int:
    count = 0
    for number in sys.argv[1:]:
        if is_prime(int(number)):
            count += 1
    return count

def main():
    print(count_of_primes())

if __name__ == '__main__':
    main()
```

Number of prime elements

```python
import sys
import math

def isprime(n: int) -> bool:
    if n == 0 or n == 1:
        return False
    d = 2
    while d <= math.sqrt(n):
        if n % d == 0:
            return False
        d += 1
    return True
```

```python
def count_of_primes(numbers: list[int]) ->
int:
    count = 0
    for number in numbers:
        if isprime(number):
            count += 1
    return count

def main():
    numbers = []
    argc = len(sys.argv)
    for i in range(1, argc):
        numbers.append(int(sys.argv[i]))

    print(count_of_primes(numbers))

if __name__ == "__main__":
    main()
```

File Handling

- **Open a file**
- To open files, you can use the **open()** function, which returns a file object.
  - **It takes the name of the file to open as the first parameter.**
  - **As a second parameter, you can specify how to open the file:**
    - read: `'r'`,
    - write: `'w'`,
    - append: `'a'`.
    - (The default value is read, i.e. 'r'.)

```
file = open("test.txt", 'r')
```

File Handling

- **Open a file**
  - To open a file for reading it is enough to specify the name of the file:

```
file = open("test.txt")
```

  - If the file is in a different location, you will have to specify the file path, like this:

```
file = open("c:/IntroProg/test.txt")
```

Read data from a file

Read the entire contents of the file:

```
with open('test.txt') as file:
    lines = file.read()
    print(lines)
```

Read data from a file

Read the first line of the file:

```
with open("test.txt") as file:
    line = file.readline().strip("\n")
    print(line)
```

Read data from a file

```
Read all lines of the file,
the result is a list of strings:

with open("test.txt") as file:
    lines = file.readlines()
    print(lines)
```

Read data from a file

Read the file line by line:

```
with open("test.txt") as file:
      for line in file:
         print(line.strip("\n"))
```

File Handling

- **Create a file**

- Writing in a file is done with the `write()` function of the file object, which works similarly to the print() function.

- Note, however, that the `write()` function can only have one string parameter, so if you want to write out the values of multiple variables on a single line, you must use concatenation or string formatting.

Writing to a file

Writing to a file – one line:

```
with open("out.txt", "w") as file:
    file.write("apple\n")
```

Writing to a file

```
Writing to a file – more line:

with open("out.txt", "w") as file:
    file.write("apple\n")
    file.writelines(["pear\n", "orange\n"])
```

Exceptions

```
try:
    with open("out3.txt") as file:
        for line in file:
            print(line.strip("\n"))
except FileNotFoundError:
    print("File not found!")
```

Greatest common divisor

- https://progcont.hu/progcont/100132/?pid=200801

Greatest common divisor

- Write a program that reads positive integers from the text file given as its first command line argument, up to several per line (in this case the numbers will be separated by a space).
- For each input line, the program writes to the standard output the **greatest common divisor** of the numbers in the line!

Greatest common divisor

- Sample Input

sample.txt -> Command Line Argument

```
60 24 72
8
15 25 35
```

- Output for Sample Input

```
12
8
5
```

Solution

```python
import sys

def gcd(a: int, b: int) -> int:
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a
    return a

def main():
    with open(sys.argv[1]) as file:
        for line in file:
            numbers = line.strip("\n").split(" ")
            num = int(numbers[0])
            for i in range(1, len(numbers)):
                num = gcd(num, int(numbers[i]))
            print(num)
if __name__ == "__main__":
    main()
```

Street length better (Utcahosszal jobb)

- https://progcont.hu/progcont/100133/?pid=200802

Street length better

- Write a program that takes the name of a text file as its first command-line argument.
  The text file contains data about cities, one line each, in the following format:
- `city_name;street_name;street_length;bus_stops_number`
- The program should sum the lengths of streets for each city, then write to the standard output the names of cities, one per line, in decreasing order based on the total street lengths.
- If the total street length is the same for multiple cities, list their names in lexicographical order on the standard output.

Street length better

- **Sample Input**

sample.txt -> **Command line argument**

  - `Debrecen;Piac utca;5.5;4`
  - `Budapest;Bartók Béla út;8.0;12`
  - `Debrecen;Bartók Béla út;6.2;7`
  - `Szeged;Zákány utca;4.2;0`

- **Output for Sample Input**
  - `Debrecen`
  - `Budapest`
  - `Szeged`

Solution

```
import sys
streets={}

with open(sys.argv[1]) as file:
    for line in file:
        data = line.strip("\n").split(";")
        #print(data)
        if data[0] in streets:
            streets[data[0]]+=float(data[2])
        else:
            streets[data[0]]=float(data[2])
    #print(streets)
for key, value in sorted(streets.items(),
                         key=lambda x: (-x[1], x[0])):
    print(key)
```

Homework 1

Number of local maximums

- https://viskillz.inf.unideb.hu/prog/#/?week=P1081&exercise=P108109c&page=sheet

Homework 1

The function

- Write a function named `count_of_local_maximums` that returns the number of items in the list given as a parameter that are greater than both of its neighbours.

Returned value

- number of elements matching the condition

Command Line Arguments:

- `12 34 12 56 5`

Output:

- `2`

Homework 2 – Formula 1

- Write a program whose first command line argument is a text file name.
- The text file contains data of **Formula 1** drivers, one per line, in the following format:

- `driver_name;race_location;number_of_lap_taken;placing`

- The program will sum up the number of laps completed per driver and then write the names of the drivers, one per line, in descending order of the sum of the number of laps completed, to the standard output!
- The names of the riders who have the same number of laps, in lexicographical order, are written to the standard output.

Homework 2

- Command Line Argument:

```
sample.txt
```

- Contents of the sample.txt file:

```
Lewis Hamilton;Melbourne;58;1
Daniel Ricciardo;Melbourne;57;6
Lewis Hamilton;Sepang;56;2
Fernando Alonso;Sepang;20;0
```

- The result of the run on standard output:

```
Lewis Hamilton
Daniel Ricciardo
Fernando Alonso
```

Homework 3

Prime selection

- https://progcont.hu/progcont/100317/?pid=201426

Homework 3 – Prime selection

- Write a program that takes the name of a text file as its first command line argument.
- Each line of the text file contains a sequence of integers, separated by exactly one space, and each line must contain at least one value.
- Your program should collect and write to the standard output, for each line, an increasing sequence of prime numbers.
- In the case where no prime is found in the line being read, the string "NOTHING" should be output!
- See the example output for the exact format!

Homework 3 – Prime selection

- Command Line Argument:

```
sample.txt
```

- Contents of the sample.txt file:

```
1 2 3 4 5 6 7 8 9
25 13 12 10 21 53
3 5 7 3 5 7 3 5 7
2 4 8 16 32
```

- The result of the run on standard output:

```
2, 3, 5, 7
13, 53
3, 5, 7
2
```