Technological School “Electronic Systems”

associated with Technical University Sofia

PUTS TEAM_NAME

{Ivo Valchev, Stanimir Bogdanov, Dimitar Terziev}

What errors were made and how could have they been avoided in the first place?

On October 21, 2014 a test took place in Software Engineering class. The results were very much unsatisfactory. Only 4 out of 24 in class A and 0 out of 10 in class B correctly solved the problem they were given. This equates to a 16.67% success rate in class A, a 0% success rate in class B and an 11.77% success rate overall. The purpose of this report is to summarize what mistakes were commonly made by students, why they occurred and how they could have been avoided in the first place.

One of the most common mistakes made was producing a CSV called not *result.csv*, as the problem statement said, but *results.csv*, *task.csv*, *students.csv*, etc. This shows that students do not pay enough attention to detail. It may be explained with the limited amount of time combined with stress and anxiety, but the fact remains that more than 30% of the students who took this test made this very simple, but detrimental error. How could have this been avoided? Unfortunately, it is up to the students and not the teacher to carefully read the problem description given.

Another extremely common pitfall were filenames. In the description of the problem it was stated that the files in the one or two given directories were to have filenames in the format *First_Last_digits.rb*. However, halfway through the test the teacher *mentioned* that there could be files with incorrect filenames. This in itself is contradictory to the problem statement. Many students ignored making checks for valid filenames and produced wrong results. And among those who did make checks, they usually did in very awkward, non-idiomatic ways with dozens of *split* method calls, exceptions, and so on. Only 3 out of 34 students used the easiest, most straightforward solution - regular expressions. This indicates unfamiliarity or lack of skills to use them. This general group of errors could have been avoided by a more accurate description of the problem and knowledge of regular expressions.

Many students cling to their tried-and-tested style of programming, reminiscent of C. Unfortunately, this leads to bad Ruby code. Every programming language has its own style, idioms, conventions, even culture. Students seem not to be aware of this. For example, some still use traditional loops, instead of iterators or write their own methods instead of using what Ruby provides out of the box. While somewhat understandable, since most students are still learning Ruby, this makes code harder to read and harder to debug.

Lack of familiarity with Ruby in general is a problem - students often call methods like *flatten* or *capitalize* without the need to do so, or do not make a distinction between a method with and and a method without an exclamation point at the end of its name.

Lack of familiarity with data structures is yet another problem - some use hashes and wonder why students with the same names magically disappear from a hash, and only one is left! Some confuse arrays of arrays with hashes and vice-versa.

The idea of indentation appears to be alien to some students. It is mind-boggling to every half-decent programmer that one could write code without any tabs or spaces whatsoever, not to mention the traditional Ruby convention of two spaces per indentation level. This leads to code

that is hard to read, hard to debug, hard to maintain, hard to extend. And nobody wants this. While on the topic of code style, it should be mentioned that students tend to give very non-descriptive names to variables, for example “short”, “el”, “o”, “asd”, “arr”, “my_hash”, etc. This may be acceptable to some extend in the context of a 30-minute long test, but this manner of programming often evolves into a habit, which is guaranteed to cause problems in the students’ programming careers further down the road.

To sum up, students have made typographical errors, logic errors, errors caused by unfamiliarity with Ruby or data structures, and writing good code seems not to be a concern to most. Knowledge of regular expressions could have greatly helped students, and reading the description of the problem carefully and following it strictly could have increased the success rate.

Appendix 1

Author: Borislav Rusinov
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_6fb3ad.rb

1. you are given an argument for a folder with files;
1.1 if there are other arguments they should be discarded
2. file names in this folder are in the form First_Last_digits.rb;
3. find all the students that have 10 letters in their first name;
4. Sort the result by Last Name DESC.
5. Produce a result in CSV format named result.csv:

                FirstName1,LastName1
                FirstName2,LastName2
                ...
                FirstNameN,LastNameN


=end
a=ARGV[0]
require 'csv'
array=[]
Dir.glob("#{a}*.*")  do |my_text_file|
        name = my_text_file.split("/").last.split(".").first.split("_")
        if name[1]!=nil && name[0].length==10
                array << name[0] + "," + name[1]
        end
end
array.sort!
array.reverse!
File.open("results.csv", "w") do |csv|
        array.each do |arg|
        csv.puts(arg)
        end
end
```

Comments:

- Program file is named in a wrong way (“Borislav_Rusinov_2_6fb3ad.rb” instead of “Borislav_Rusinov_1_6fb3ad.rb”.
- Module ‘csv’ required, but never used (output file produced via the file module)

- Output file is named in a wrong way (“results.csv” instead of “result.csv”)
- Array data structure is not appropriate for this kind of task (it makes it harder to handle FirstName and SecondName separately)
- Code is readable

Rate: 2

Appendix 2

Author: Denis Trenchev
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_b4c3f5.rb
1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. file names in this folders are in the form First_Last_digits.rb;
3. find  the students with 5 letters in the first name that are in both folders. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
4. Sort the result by Last name ;
5. Produce a result in CSV format named result.csv:

        LastName1,FirstName1
        LastName2,FirstName2
        ...
        LastNameN,FirstNameN

=end

require 'csv'

i = 0
arr1 = []
arr2 = []
arr3 = []

Dir.glob(ARGV[0]+"*.rb") do |first_folder|
        name = first_folder.split('/').last.split('.').first.split('_')

        if name.length == 3
                if name[1].to_s.length == 5
                        arr1[i] = []
                        arr[i][0] = name[0]
                        arr[i][1] = name[1]
                        i+=1
                end
        end
end
i = 0
```

```
Dir.glob(ARGV[1]+"*.rb") do |second_folder|
        name = second_folder.split('/').last.split('.').first.split('_')

        if name.length == 3
                if name[1].to_s.length == 5
                        arr1[i] = []
                        arr[i][0] = name_1[0]
                        arr[i][1] = name_1[1]
                        i+=1
                end
        end
end
i = 0

arr1.each do |compare1|
        arr2.each do |compare2|
                if compare2 == compare1
                        arr3[i] = compare1
                        i+=1
                end
        end
end


sort = arr3.sort_by{|asd| asd[1]}
CSV.open("students.csv", "w") do |csv|
   sort.each do |element|
      csv << element
   end
end
```

Comments:

- Inefficiently using arrays and memory
- Non-descriptive variables (asd, compare1, compare2)
- Using non-declared variables
- Code is not readable

Rate: 1

Appendix 3

Program author: Dimitar Nesterov
Code:

```
#Develop a program named FirstName_LastName_ClassNumber_0d5526.rb
#
#1. you are given an argument for a folder with files;
#1.1 if there are other arguments they should be discarded
#2. file names in this folder are in the form First_Last_digits.rb;
#3. find all the students that have 10 letters in their first name;
#4. Sort the result by Last Name DESC.
#5. Produce a result in CSV format named result.csv:
#
#               FirstName1,LastName1
#               FirstName2,LastName2
#               ...
#               FirstNameN,LastNameN


require 'csv'
def is_numeric(o)
   true if Integer(o) rescue false
end
array = []
count = 0
Dir.glob(ARGV[0] + "*.rb") do |file|
        name = file.split("/").last.split(".").first.split("_")

        name[0] = name[0].to_s
        name[0] = name[0].capitalize

        name[1] = name[1].to_s
        name[1] = name[1].capitalize

        if name.size == 3 && is_numeric(name[2])
                if name[1].length == 10

                        array[count] = []
                        array[count][0] = name[0].to_s
                        array[count][1] = " #{name[1].to_s}"
                        count += 1

                end
```

```
        end
end
array = array.sort_by {|el| -el[1]}
CSV.open("result.csv", "w") do |csv|

                array.uniq.each do |e|

                        csv << e

                end

end
```

Comments:

- Losing data by using capitalize
- Non-descriptive variables (o,el)
- Code is relatively readable

Rate: 2

Appendix 4

Author: Dimitar Terziev
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_88db52.rb

1. you are given an argument for a folder with files;
1.1 if there are other arguments they should be discarded
2. file names in this folder are in the form First_Last_digits.rb;
3. find all the students that have 5 letters in their second name;
4. Sort the result by Last Name ASC.
5. Produce a result in CSV format named result.csv:

                FirstName1,LastName1
                FirstName2,LastName2
                ...
                FirstNameN,LastNameN


=end
require 'csv'
arr = []
Dir.glob("#{ARGV[0]}*.rb*"){|file|
        file_str = file.split('/').last
        if(file_str=~/\A[a-zA-Z]+\_[a-zA-Z]+\_\d+\.rb\z/ && file_str.split('_')[1].size == 5)
                arr.push("#{file_str.split('_')[1]} #{file_str.split('_').first}")
        end
}
CSV.open('result.csv','w'){|csv|
        arr.uniq.sort.each{|el|
                csv << "#{el.split(' ').last} #{el.split(' ').first}".split(' ')
        }
}
```

Comments:

- Code is not readable
- Non-descriptive variables (el)

Rate: 4

Appendix 5

Author: Ivelin Slavchev
Code:

```
=begin
        Develop a program named FirstName_LastName_ClassNumber_835552.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. Find all the files from both folders that are not in the format FirsrName_LastName_digits.rb
If there are duplicates the file must be written only once. If two files are of the same lenght
those files should be sorted in ASC order;
3. Calculate the length of their names (including extensions).;
4. Sort the result by lenth ;
5. Produce a result in CSV format named result.csv:

                        File1,3
                        File2,4
                        ...
                        FileN,3
=end

require 'csv'
result = Hash.new
Dir.glob(ARGV[0] + "*").each do |file1|
        short1 = file1.split("/").last
        ext1 = short1.split(".").last
        names1 = short1.split(".").first
        digit1 = file1.split("_").last
        if (ext1 != "rb") or (digit1.to_i.to_s != digit1) or (short1.scan("_").count != 2)
                result[short1] = short1.length
        end
end
Dir.glob(ARGV[1] + "*").each do |file2|
        short2 = file2.split("/").last
        ext2 = short2.split(".").last
        names2 = short2.split(".").first
        digit2 = file2.split("_").last
        if (ext2 != "rb") or (digit2.to_i.to_s != digit) or (short2.scan("_").count != 2)
                result[short2] = short2.length
        end
end
result.sort_by{|k, v| v}
```

```
CSV.open("result.csv", "w") do |csv|
        result.each do |p|
                csv << p
        end
end
```

Comments:

- Using non-declared variables (digit)
- Some problems are not catched (two files with same length should be sorted in ASC order)
- Code is readable

Rate: 3

Appendix 6

Author: Ivo Valchev
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_6c8bd9.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. file names in this folders are in the form First_Last_digits.rb;
3. find  the students with 5 letters in the first name that are in both folders. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
4. Sort the result by Last name ;
5. Produce a result in CSV format named result.csv:

        LastName1,FirstName1
        LastName2,FirstName2
        ...
        LastNameN,FirstNameN
=end

hash_fold1={}
hash_fold2={}

Dir.glob("#{ARGV[0]}*.*")  do |file|
                name = file.split("/").last.split(".").first.split("_")
                isNum = Integer(name[2]) rescue nil
                if name[0] and name[1] and name[0].length == 5 and !isNum!=nil
hash_fold1.include?(name[0])
                        hash_fold1["#{name[1]}"] = "#{name[0]}"
                end
end
Dir.glob("#{ARGV[1]}*.*") do |file|
                name = file.split("/").last.split(".").first.split("_")
                isNum = Integer(name[2]) rescue nil
                if name[0] and name[1] and name[0].length == 5 and !isNum!=nil
and!hash_fold2.include?(name[0])
                        hash_fold2["#{name[1]}"] = "#{name[0]}"
                end
end
File.open("result.csv", "w") do |csv|
        hash_fold1.sort.map do |key, value|
                if (hash_fold1[key]==hash_fold2[key])
```

```
                    csv.puts("#{key},#{value}")
                end
            end
        end
```

Comments:

- Wrong condition (!isNum!=nil)
- Program can be optimized to use one hash instead of two
- Code is relatively readable

Rate: 2

Appendix 7

Author: Kalin Marinov
Code:

```
#==begin
#Develop a program named FirstName_LastName_ClassNumber_bce70c.rb
#
#1. you are given an argument for a folder with files;
#1.1 if there are other arguments they should be discarded
#2. file names in this folder are in the form First_Last_digits.rb;
#3. find all the students that have 5 letters in their second name;
#4. Sort the result by First name DESC.
#5. Produce a result in CSV format named result.csv:
#
#               FirstName1,LastName1
#               FirstName2,LastName2
#               ...
#               FirstNameN,LastNameN
#==end

require 'csv'

hash = Hash.new

Dir.glob("#{ ARGV[0] }/*") do |name|
        name = name.split("/").last
        short_name = name.split('_')[1]
        if short_name.length == 5
                hash[name] = short_name
        end
end

CSV.open("result.csv", "w") do |csv|
        hash = hash.sort_by { |key, value| value }.reverse
        hash.each |key| do
                csv << key
        end
end
```

Comments:

- Invalid ruby syntax (hash.each |key| do)
- No checks for file form (short_name !=nil ?)
- Code is readable

Rate: 2

Appendix 8

Author: Kamena Dacheva
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_0af18f.rb

1. you are given an argument for a folder with files;
1.1 if there are other arguments they should be discarded
2. file names in this folder are in the form First_Last_digits.rb;
3. find all the students that have 5 letters in their second name;
4. Sort the result by First name DESC.
5. Produce a result in CSV format named result.csv:

                FirstName1,LastName1
                FirstName2,LastName2
                ...
                FirstNameN,LastNameN
=end

student = Hash.new { |name, programs| name[programs] = []}
directory = ARGV[0]
require "csv"

class String
  def is_number?
    Float(self) != nil rescue false
  end
end

Dir.glob("#{directory}/*.*") do |my_repository|

        name_dir = my_repository.split("/").last

        name = name_dir.split("_").first.capitalize
        sir_name = name_dir.split("_", 2).last.split("_").first.capitalize
        program = name_dir.split("_").last.split(".").first
        ex = name_dir.split("_").last.split(".").last

        if name_dir.include? "_" then counter = name_dir.count "_" end
        student["#{name}"] << sir_name if ((counter == 2) && (sir_name.length == 5) &&
(program.is_number?) && (ex == "rb"))
end
```

```ruby
CSV.open("result.csv", "w") do |csv|
        student.sort_by{|k, v| v}.reverse.each do |f_name, l_name|
                csv << [f_name,l_name].flatten
        end
end
```

Comments:

- Wrong file path (“#{directory}/*.*” instead of “#{directory}*.*”)
- Sorting by LastName, instead of FirstName
- Code is relatively readable

Rate: 3

Appendix 9

Author: Kristina Pironkova
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_890ba0.rb

1. you are given an argument for a folder with files;
1.1 if there are other arguments they should be discarded
2. file names in this folder are in the form First_Last_digits.rb;
3. find all the students that have 10 letters in their first name;
4. Sort the result by Last Name DESC.
5. Produce a result in CSV format named result.csv:

                FirstName1,LastName1
                FirstName2,LastName2
                ...
                FirstNameN,LastNameN
=end


require 'csv'
results=Hash.new
Directory = ARGV[0]
Dir.glob("#{Directory}/*.rb") do |file_name|

        first_name = file_name.split("/").last.split("_").first.capitalize
        last_name=file_name.split("/").last.split("_",2).last.split("_").first.capitalize

                if first_name.length == 10

                        results["#{last_name}"] ="#{first_name}"
                end

end


CSV.open("results.csv", "w") do |csv|
        results.sort.each do |first,last|

        csv << [last,first]

        end
```

```
end
```

Comments:

- Wrong file path (“#{Directory}/*.rb” instead of “#{Directory}*.rb”)
- Sorting in ASC instead of DESC order
- Output file is named in a wrong way (“results.csv” instead of “result.csv”)
- Code is readable

Rate: 1

Appendix 10

Program author: Lubomir Yankov
Code:

```
require 'csv'
def is_numeric(o)
   true if Integer(o) rescue false
end

array = []
count = 0

Dir.glob(ARGV[0] + "*").each do |file|
        ch_count = 0
        file_name = file.split("/").last.split("")

        file_name.each do |ch|

                if is_numeric(ch)

                        ch_count += 1

                end

        end
        if ch_count == 9
                len = file_name.length
                array[count] = []
                array[count][0] = file_name
                array[count][1] = len/2.round
                count += 1
        end

end
array = array.sort_by {|el| el[0]}
CSV.open("results.csv", "w") do |csv|

        array.each do |element|

                csv << element

        end
```

```
end
```

Comments:

- Missing task description

Rate: NIL

Appendix 11

Author: Marian Belchev
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_ad26e0.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. file names in this folders are in the form First_Last_digits.rb;
3. find  the students that are only in the second folder and not in the first. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
4. Sort the result by First name ;
5. Produce a result in CSV format named result.csv:

        LastName1,FirstName1
        LastName2,FirstName2
        ...
        LastNameN,FirstNameN
=end

require 'csv'

hash1 = Hash.new
hash2 = Hash.new

Dir.glob("#{ARGV[0]}*_*_*.rb") do |file1|
        Dir.glob("#{ARGV[1]}*_*_*.rb") do |file2|
                firstName1     = file1.split("/").last.split("_").first
                lastName1      = file1.split("/").last.split("_", 2).last.split("_").first
                number1 = file1.split("_").last.split(".").first

                firstName2     = file2.split("/").last.split("_").first
                lastName2      = file2.split("/").last.split("_", 2).last.split("_").first
                number2 = file2.split("_").last.split(".").first

                hash1[firstName1] = lastName1 + "." + number1
                hash2[firstName2] = lastName2 + "." + number2
        end
end

CSV.open("results.csv", "w") do |csv|
```

```
        hash2.sort.each do |key, value|
                if !hash1.has_key?(key) && !hash1.has_value?(value.split(".").first) &&
!hash1.has_value?(value.split(".").last.to_i)
                                csv << [key,value.gsub('.',"")]
                end
                if hash1.has_key?(key) && !hash1.has_value?(value.split(".").first) &&
!hash1.has_value?(value.split(".").last.to_i)
                        csv << [key,value.gsub('.',"")]
                end
        end
end
```

Comments:

- Writes to *results.csv* instead of *result.csv*
- Not efficient - has nested Dir.glob methods
- The fragment for writing data to a CSV file is complicated, unreadable and incorrect
- The output format is FirstName,LastNameNumber instead of FirstName,LastName
- Does not sort correctly
- Correctly traverses the two given directories

Rate: 2

Appendix 12

Author: Momchil Angelov
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_d8aa65.rb

1. you are given two arguments for a folders with files;
1.1 If there are other arguments they should be discarded;
2. Find all the files from both folders that are not in the format FirsrName_LastName_digits.rb
If there are duplicates the file must be written only once.
2.1 If two files are of the same lenght those files should be sorted in ASC order;
3. Calculate the length of their names (including extensions).;
4. Sort the result by lenth ;
5. Produce a result in CSV format named result.csv:

                        File1,3
                        File2,4
                        ...
                        FileN,3
=end
require 'csv'

arr1=Array.new
arr2=Array.new
arr3=Array.new
a = ARGV[0]
b = ARGV[1]
i=0
Dir.glob(a + "/*.rb") do |my_text_file1|
        short= my_text_file1.split('/').last
        length1 = short.length
        shorter= short.split('.').first.split('_')
        first_name=shorter[0]
        last_name=shorter[1]
        digits=shorter[2].to_i


        if !first_name || !last_name || digits=0
                next
        else
                arr1 << ["#{short}" "#{length1}"]
```

```
        end
end
Dir.glob(b + "/*.rb") do |my_text_file2|

        short2= my_text_file2.split('/').last
        length2 = short2.length
        shorter2= short.split('.').first.split('_')
        first_name2=shorter2[0]
        last_name2=shorter2[1]
        digits2=shorter2[2].to_i


        if !first_name2 || !last_name2 || digits2=0
                next
        else
                arr2 << ["#{short2}","#{length2}"]
        end
end

        arr3 = arr1 & arr2



        arr3 = arr3.sort_by  {|el|
                 el[1]
        }



    CSV.open("result.csv", "w") do |csv|

arr3.each do |element|
csv << element
end

end
```

Comments:

- Error on line 37: *arr1 << ["#{short}" "#{length1}"]* -> no comma
- Does not produce a result.csv
- Not very descriptive variable names (e.g. short, shorter, short2)
- Calls the variable short which is out of scope instead of short2

- Uses = instead of == for integer comparison
- With all this fixed it still produces an incorrect result

Rate: 3

Appendix 13

Author: Nikola Marinov
Code:

```
=begin
1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. Find all the files from both folders that have exactly 7 digits from 0 to 9 in their names
excluding extension. If there are duplicates the file must be written only once.;
3. Calculate the length of their names (including extensions) divided by 2 rounded to the
smalles number;
4. Sort the result by File name ;
5. Produce a result in CSV format named result.csv:

                        File1,3
                        File2,4
                        ...
                        FileN,3
=end

requre 'csv'
def is_numeric(o)
 true if Integer(o) rescue false
 end

 array=[]

count=0
Dir.glob(ARGV[0] + "/**/*.*").each do |file|

 full_name=file.split("/").last
 name = file.split("/").last.split(".").first_split("_")

 if name.lenght != 3 && !is_numeric(name[2])
 array(count) = []
 array(count) [0]=full_name
 array(count)[1]= full_name.to_s.lenght
 count += 1

 end
 end
```

```
Dir.glob(ARGV[0] + "/**/*.*").each do |file|

full_name=file.split("/").last
name = file.split("/").last.split(".").first_split("_")

if name.lenght != 3 && !is_numeric(name[2])
array(count) = []
array(count) [0]=full_name
array(count)[1]= full_name.to_s.lenght
count += 1
end
end
array = array.sort_by{|el| el|0|}

CSV.open("task.csv",w) do |csv|
array=uniq.each do |element|
csv << element
end
end
```

Comments:

- Writes to task.csv instead of result.csv
- Has typographical errors - requre instead of require, lenght instead of length, etc.
- Horrendously unformatted - tabs appear not to be a known concept to the author of this program
- Error on line 29: *array(count) = []* -> syntax error
- Error on line 51: *CSV.open("task.csv",w) do* … -> *w* is not quoted

Rate: 1

Appendix 14

Author: Petko Bozhinov
Code:

```
# Develop a program named FirstName_LastName_ClassNumber_954dc6.rb

# 1. you are given two arguments for a folders with files;
# 1.1 if there are other arguments they should be discarded;
# 2. file names in this folders are in the form First_Last_digits.rb;
# 3. find  the students with 5 letters in the first name that are in both folders. A student is in
both folders if it there is a file with the same First and Last Name. Digits might be different;
# 4. Sort the result by Last name ;
# 5. Produce a result in CSV format named result.csv:

#       LastName1,FirstName1
#       LastName2,FirstName2
#       ...
#       LastNameN,FirstNameN

require 'csv'

class String
  def numeric?
    Float(self) != nil rescue false
  end
end

output = Array.new
i = 0
Dir.glob(ARGV[0] + "/*") do |file|
        file = file.split('/').last.split('.').first.split('_')
        Dir.glob(ARGV[1] + "/*") do |file2|
                file2 = file2.split('/').last.split('.').first.split('_')
                if "#{file[0]} #{file[1]}" == "#{file2[0]} #{file2[1]}"
                        if file[2].numeric?
                                if file[0].to_s.length == 5
                                        output[i] = Array.new
                                        output[i][0] = file[0]
                                        output[i][1] = file[1]
                                        i+=1
                                end
                        end
```

```
                end
        end
end

output = output.sort_by{ |element| element[1]}
CSV.open("result.csv", "w") do |csv|
        output.each do |pusher|
                csv << pusher
        end
end
```

Comments:

- Not efficient - has nested Dir.glob methods
- Added the method *numeric?* to the *String* class which is interesting
- I do not know why the official results say no *result.csv* was found, because this program **does** produce a *result.csv* file.
- The output format is FirstName,LastName instead of LastName,FirstName
- Overall very close

Rate: 5

Appendix 15

Author: Radoslav Kostadinov
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_772118.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. file names in this folders are in the form First_Last_digits.rb;
3. find  the students that are only in the second folder and not in the first. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
4. Sort the result by First name ;
5. Produce a result in CSV format named result.csv:

        LastName1,FirstName1
        LastName2,FirstName2
        ...
        LastNameN,FirstNameN

=end

require 'csv'
file1 = Hash.new
file2 = Hash.new

path1 = ARGV[0]
path2 = ARGV[1]

Dir.glob("#{path1}*.rb") do |my_text_file|
                s = my_text_file.split(/\//).last.capitalize
                first_name = my_text_file.split("/").last.split("_").first
                last_name  = my_text_file.split("/").last.split("_",2).last.split("_").first


                if s.count('_') == 2 and !((first_name == "" || first_name == " ") || (last_name
== "" || last_name == " "))
                                file1[first_name] = last_name
                        end
end

Dir.glob("#{path2}*.rb") do |my_text_file|
```

```
                s = my_text_file.split(/\//).last.capitalize
                first_name = my_text_file.split("/").last.split("_").first
                last_name  = my_text_file.split("/").last.split("_",2).last.split("_").first

                if s.count('_') == 2 and !((first_name == "" || first_name == " ") || (last_name
== "" || last_name == " "))
                                file2[first_name] = last_name
                        end
end

CSV.open("result.csv", "w") do |csv|
        file1.sort.each do |first_name, last_name|
                file2.sort.each do |first_name1, last_name1|
                 if first_name1 == first_name and last_name1 == last_name
                        begin
                        end
                        else
                                csv << [last_name1, first_name1]
                        end
                end
        end
        end
```

Comments:

- Has very long lines up to 124 characters
- Correctly traverses the two given directories
- Does not do what the problem statement says
- Problems with conditional statements - if, begin, end, else, end…? (refer to the section for writing to a CSV file)

Rate: 1

Appendix 16

Author: Simeon Shopkin
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_56a835.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. Find all the files from both folders that are not in the format FirsrName_LastName_digit.rb.
there are duplicates the file must be written only once. If two files are of the same lenght thos
files should be sorted in ASC order;
3. Calculate the length of their names (including extensions).;
4. Sort the result by length ;
5. Produce a result in CSV format named result.csv:

                        File1,3
                        File2,4
                        ...
                        FileN,3

=end

require 'csv'

arr = Array.new
        Dir.glob(ARGV[0]+"/*.rb") do |first_files|
                Dir.glob(ARGV[1]+"/*.rb") do |second_files|
                        first_files = first_files.split("/").last.split(".").first.split("_")
                        if first_files.size != 3
                                if first_files != second_files
                                                print_count = first_files.split("/").last.split(".").first

                                                p = print_count.size.to_s
                                                print =
first_files[0].capitalize+"_"+first_files[1].capitalize+"_"+first_files[2]+","+p
                                                arr.push(print)
                                end
                        end
                end
        end
```

```
CSV.open("result.csv","w") do |csv|
        arr.sort.each do |element|
                csv << [element]
        end
end
```

Comments:

- first_files is a String at first, then it becomes an array but String methods like split are still called upon it (e.g. line 26)
- Has nested Dir.glob calls when two consecutive Dir.glob calls is the way to go
- Does a lazy format check without regular expressions
- Does not work

Rate: 1

Appendix 17

Author: Stanislav Gospodinov
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_b36abb.rb

1. you are given an argument for a folder with files;
1.1 if there are other arguments they should be discarded
2. file names in this folder are in the form First_Last_digits.rb;
3. find all the students that have 5 letters in their second name;
4. Sort the result by Last Name ASC.
5. Produce a result in CSV format named result.csv:

                FirstName1,LastName1
                FirstName2,LastName2
                ...
                FirstNameN,LastNameN
=end

require 'csv'
hash = Hash.new

Dir.glob("#{ARGV[0]}*.rb") do |file|
        filename = file.split('/').last.split('.').first;
                if filename.split('_').length == 3
                        if filename.split('_')[1].length == 5
                                hash[filename.split('_')[0]] = filename.split('_')[1]
                        end
                end
end

hash = Hash[hash.sort_by{|k, v| v}]

CSV.open("results.csv", "w") do |csv|
        hash.each do |key, value|
                csv << [key, value].flatten
        end
end
```

Comments:

- Writes to *results.csv* instead of *result.csv*
- The program is easier than most others
- Works correctly
- Needlessly calls flatten on line 33

Rate: 5

Appendix 18

Author: Stanislav Valkanov
Code:

```
#Develop a program named FirstName_LastName_ClassNumber_4482c1.rb

#1. you are given an argument for a folder with files;
#1.1 if there are other arguments they should be discarded
#2. file names in this folder are in the form First_Last_digits.rb;
#3. find all the students that have 5 letters in their second name;
#4. Sort the result by First name DESC.
#5. Produce a result in CSV format named result.csv:

#               FirstName1,LastName1
#               FirstName2,LastName2
#               ...
#               FirstNameN,LastNameN

require 'csv'
a = Hash.new
path = ARGV[0]
Dir.glob(path + "**/*.rb") do |my_text_file|
short_name = my_text_file.split('/').last.split('.').first
name = short_name.split("_")[0]
last = short_name.split("_")[1]
last.to_s
if (last.length == 5)&&(short_name.split("_").size == 3)
a["#{name}"] = last
end
end
CSV.open("result.csv", "w") do |csv|
Hash[a.sort.reverse].each do |element|
csv << element
end
end
```

Comments:

- The program is easier than most others
- Does a fairly lazy check for correct format so the program doesn't work with files like oneword.rb

- Horrendously unformatted - tabs appear not to be a known concept to the author of this program

Rate: 3

Appendix 19

Program author: Tihomir Lidanski
Code:

```
#Develop a program named FirstName_LastName_ClassNumber_dafd44.rb

#1. you are given two arguments for a folders with files;
#1.1 if there are other arguments they should be discarded;
#2. Find all the files from both folders that have exactly 7 digits from 0 to 9 in their names
excluding extension. If there are duplicates the file must be written only once.;
#3. Calculate the length of their names (including extensions) divided by 2 rounded to the
smalles number;
#4. Sort the result by File name ;
#5. Produce a result in CSV format named result.csv:

#                       File1,3
#                       File2,4
#                       ...
#                       FileN,3

require 'csv'

Dir.glob(ARGV[0] + "*.") do |file|
        name = file.split ("/")last.split(".")


Dir.glob(ARGV[1] + "*.") do |file|




puts name.length % 2.round()



end
end

CSV.open("result.csv", "w") do |csv|
```

```
end
```

Comments:

- No comment - this piece of code is a skeleton for a program at best

Rate: -1

Appendix 20

Author: Veselin Dechev
Code:

```
require 'csv'
result = Hash.new
Dir.glob(ARGV[0] + "*.rb").each do |first|
        name1 = first.split("/").last.capitalize
        first_name = name1.split("_").first.capitalize
        last_name = name1.split("_",2).last.split('_').first.capitalize
        Dir.glob(ARGV[1]+"*.rb").each do |second|
                name2 = second.split("/").last.capitalize
                if (name1 == name2)
                        result.compare_by_identity
                        result[first_name] = last_name
                end
end
end
CSV.open("result.csv", "w") do |csv|
        result.sort_by{|k, v| k}.each do |element|
                csv << element
                end
        end
```

Comments:

- Wrong filename: Veselin_Dechev_11A2_5f1c22.rb
- The program file does not include the problem statement
- The author seems to have misunderstood the problem statement - it asks for students found in the first folder but not the second, while the program works with the students found in both folders

Rate: 3

Appendix 21

Author: Borislav Stratev
Code:

```ruby
#Develop a program named FirstName_LastName_ClassNumber_a65be5.rb

#1. you are given two arguments for a folders with files;
#1.1 if there are other arguments they should be discarded;
#2. file names in this folders are in the form First_Last_digits.rb;
#3. find  the students that are only in the first folder and not in the second. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
#4. Sort the result by Last name ;
#5. Produce a result in CSV format named result.csv:

#       LastName1,FirstName1
#       LastName2,FirstName2
#       ...
#       LastNameN,FirstNameN

require 'csv'
a = Array.new
h = Hash.new
Dir.glob("#{ARGV[0]}/*.rb") do |dir_file_name_1|
        Dir.glob("#{ARGV[1]}/*.rb") do |dir_file_name_2|

                file_name_1 = dir_file_name_1.split(/\//).last.to_s
                file_name_2 = dir_file_name_2.split(/\//).last.to_s

                if(file_name_1 != file_name_2)
                        file_name = file_name_1
                        digit = file_name.split(/_/).last.split(/\./).first.to_s
                        first_name = file_name.split(/_/).first.to_s
                        full_first_name = first_name + digit
                        full_first_name = full_first_name.to_s
                        tmp = file_name.split("#{first_name}_")
                        full_last_name = tmp.last.split(/_/).first.to_s + digit
                        full_last_name = full_last_name.to_s
                        h[full_last_name] = full_first_name

                end
        end
end
```

```ruby
CSV.open("results.csv", "w") do |csv|
        a = h.sort
        a.each do |element|
                csv << element
        end
end
```

Comments:

- No file name checks
- Wrong output file name("results.csv")
- Many variables and unnecessary actions(not critical but worth notice)
- Misunderstood problem description leading to false goal

Rate: 2

Appendix 22

Author: David Georgiev

Code:

```ruby
        #Develop a program named FirstName_LastName_ClassNumber_1eea4f.rb

#1. you are given an argument for a folder with files;
#1.1 if there are other arguments they should be discarded
#2. file names in this folder are in the form First_Last_digits.rb;
#3. find all the students that have 5 letters in their second name;
#4. Sort the result by Last Name ASC.
#5. Produce a result in CSV format named result.csv:

#               FirstName1,LastName1
#               FirstName2,LastName2
#               ...
#               FirstNameN,LastNameN

        require 'csv'
        students_names = []
        Dir.glob("#{ARGV[0]}/**/*.rb") do |current_file|

        name = current_file.split('/').last.split(/_/)
        if name[1].length == 5
                if not students_names.include?(["#{name[1]}", "#{name[0]}"]) then
                        students_names << (["#{name[1]}", "#{name[0]}"])
                end
        end
        end
        CSV.open("result.csv", "w") do |csv|
                students_names.sort.each do |last, first|
                        csv << ["#{first}",  "#{last}"]
                end
        end
```

Comments:

- no checks for correct filename
- error in .length method because the author thought the file iterator to be string(and it isn't)
- short code with no/little unnecessary actions

Quickfix: 20:if name[1].length == 5 -> if name[1].to_s.length == 5

Rate: 4

Appendix 23

Author: Iliyan Germanov
Code:

```
=begin
        Develop a program named FirstName_LastName_ClassNumber_f8b0d9.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. file names in this folders are in the form First_Last_digits.rb
3. find  the students that are only in the first folder and not in the second. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
4. Sort the result by Last name ;
5. Produce a result in CSV format named result.csv:

        LastName1,FirstName1
        LastName2,FirstName2
        ...
        LastNameN,FirstNameN
=end

require 'csv'
results = Hash.new
results.compare_by_identity
def is_number(str)
        str[/[0-9]+/] == str
end
Dir.glob("#{ARGV[0]}/*.rb") do |path1|
        filename1 = path1.split(/\//).last
        if filename1.count("_") == 2
                firstname1 = filename1.split("_").first
                lastname1 = filename1.split("_")[1]
                digit1 = filename1.split("_")[2].split(".").first
                if is_number(digit1)
                        flag = 0
                        Dir.glob("#{ARGV[1]}/*.rb") do |path2|
                                filename2 = path2.split(/\//).last
                                if filename2.count("_") == 2
                                        digit2 = filename2.split("_")[2].split(".").first
                                        if is_number(digit2)
                                                name1 = firstname1 + lastname1
                                                name2 = filename2.split("_").first +
```

```
filename2.split("_")[1]
                                                    if name1 == name2
                                                            flag = 1
                                                            break
                                                    end
                                            end
                                    end
                            end
                            if flag == 0
                                    results[lastname1] = firstname1
                            end
                    end
            end
end

CSV.open("result.csv", "w") do |csv|
        results.sort_by{|key, val| key}.each do |el|
                csv << el
        end
end
```

Comments:

- Not through enough checks
- Demonstrated skills but in useless code(the num checking method)

Rate: 4

Appendix 24

Author: Lili Karakoleva
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_e0ea9c.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. file names in this folders are in the form First_Last_digits.rb;
3. find  the students that are only in the second folder and not in the first. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
4. Sort the result by First name ;
5. Produce a result in CSV format named result.csv:

        LastName1,FirstName1
        LastName2,FirstName2
        ...
        LastNameN,FirstNameN
=end

require 'csv'
student = Array.new
student1 = Array.new

Dir.glob(ARGV[0]+"/**/*.*").each do |file_name1|
        file_name = file_name1.split("/").last
        first_name = file_name.split("/").last.split("_").first
        p first_name
        last_name = file_name.split("/").last.split("_",2).last.split("_").first
        #task = file_name.split("_").last.split(".").first
        student << ["#{first_name}", "#{last_name}"]
end

Dir.glob(ARGV[1]+"/**/*.*").each do  |file_name1|
        file_name = file_name1.split("/").last
        first_name = file_name.split("/").last.split("_").first
        p first_name
        last_name = file_name.split("/").last.split("_",2).last.split("_").first
        #task = file_name.split("_").last.split(".").first
        student1 << ["#{first_name}", "#{last_name}"]
end
```

```
CSV.open("result.csv", "w") do |csv|
        student.each do |fn, ln|
                student1.each do |fn1, ln1|
                        if fn != fn1
                                if ln != ln1
                                        csv << ["#{fn1}", "#{ln1}"]
                                end
                        end
                end
        end
end


student1.sort.uniq.each do |fn, ln|
        ch = 0
        student.sort.uniq.each do |fn1, ln1|
                if fn == fn1 && ln == ln1
                        ch = 1
                end
        end
        if ch == 0
                puts "#{fn},#{ln}"
                csv << ["#{fn}", "#{ln}"]
        end
end
```

Comments:

- Left extensions,no sorting,no duplicates removal
- Messed up writing to csv
- Readable code but with some unnecessary actions

Quickfix: last_name(_1).split('.').first

38:student1.each do |fn, ln| -> student.sort.uniq.each do |fn, ln|

40:student.each do |fn1, ln1| -> student1.sort.uniq.each do |fn1, ln1|

checker (ch=0) after student.each becoming 1 if theres a file matching it at 40. and writing if it's 0

Rate: 3

Appendix 25

Program Author:Nikolay_Mihailov
Code:

```
#Develop a program named FirstName_LastName_ClassNumber_f70059.rb

#1. you are given two arguments for a folders with files;
#1.1 if there are other arguments they should be discarded;
#2. Find all the files from both folders that have exactly 7 digits from 0 to 9 in their names
excluding extension. If there are duplicates the file must be written only once.;
#3. Calculate the length of their names (including extensions) divided by 2 rounded to the
smalles number;
#4. Sort the result by File name ;
#5. Produce a result in CSV format named result.csv:

#                       File1,3
#                       File2,4
#                       ...
#                       FileN,3

require 'csv'
hash = Hash.new
count = 0
        Dir.glob(ARGV[0] + "/*.rb") do |file|

                first = file.split(/\//).last
                puts first

                #for (i = 0;i < first.length;i+=1)
                size = first.length
                i = 0
                first.each do |element|

                        c = first[i].chr
                        if element == 0 || element == 1 || element == 2 || element == 3 || elemer
== 4 || element == 5 || element == 6 || element == 7 || element == 8 || element == 9
                        count +=1

                        end
                end
                puts count
        end
```

```
        Dir.glob(ARGV[1] +"/*.rb") do |secFile|
                sec = secFile.split(/\//).last
                #puts sec

        end

        CSV.open("result.csv", "w") do |csv|
                hash.sort_by{|key,val| key}.each do |element|
                csv << element
                end
        end
```

Comments:

- too C
- not compiling
- not finished

Rate: 1

Appendix 26

Author: Stanislav Iliev
Code:

```
#Develop a program named FirstName_LastName_ClassNumber_627d43.r#
#
#1. you are given two arguments for a folders with files;
#1.1 if there are other arguments they should be discarded;
#2. file names in this folders are in the form First_Last_digits.rb;
#3. find  the students that are only in the first folder and not in the second. A student is in both
folders if it there is a file with the same First and Last #Name. Digits might be different;
#4. Sort the result by Last name ;
#5. Produce a result in CSV format named result.csv:
#
#       LastName1,FirstName1
#       LastName2,FirstName2
#       ...
#       LastNameN,FirstNameN



require 'csv'
name_array = Array.new()
name_array2 = Array.new()
support_array = Array.new()
support_array2 = Array.new()
i = 0
dir1 = ARGV[0]
dir2= ARGV[1]

Dir.glob("#{dir1}/*.*") do |file|
        name_array[i] = file.split(/\//).last
        i += 1
end
count = i
i = 0
Dir.glob("#{dir2}/*.*") do |file2|
        name_array2[i] = file2.split(/\//).last
        i += 1
end
i = 0
for check in i..count
```

```
        if name_array[check] != name_array2[check]
                        support_array[i] = name_array[check]
                        support_array2[i] = name_array2[check]
                        i += 1
                        puts support_array
                        puts support_array2
                        CSV.open("result.csv", "w") do |csv|
                                support_array.each do |element|
                                        csv << [element]
                                end
                        end
                        CSV.open("result.csv", "w") do |csv|
                                support_array2.each do |element2|
                                                csv << [element2]
                                end
                        end
        end
end
```

Comments:

- No checks for acceptable file
- Too C
- Checks are made only from one file to one other file. Impossible to determine if theres match without checking against all
- Two consequential openings for writing of the same file with file mode 'w'. Overwriting occurs on every opening so part of the data is lost

Rate: 2

Appendix 27

Author: Stefan Iliev
Code:

```
#Develop a program named FirstName_LastName_ClassNumber_d77aee.rb
#
#1. you are given two arguments for a folders with files;
#1.1 if there are other arguments they should be discarded;
#2. Find all the files from both folders that are not in the format FirsrName_LastName_digit.rb
If there are duplicates the file #must be written only once. If two files are of the same lenght
those files should be sorted in ASC order;
#3. Calculate the length of their names (including extensions).;
#4. Sort the result by length ;
#5. Produce a result in CSV format named result.csv:
#
#                       File1,3
#                       File2,4
#                       ...
#                       FileN,3

require 'csv'

first_folder = ARGV.shift
second_folder = ARGV.shift || "err"
names_hash = Hash.new

Dir.glob(first_folder+"/*.*").each do |text_file|
        text_file = text_file.split("/").last
        if (text_file.split("_").length == 3) then
                first_name = text_file.split("_")[0]
                second_name = text_file.split("_")[1]
                diggit = text_file.split("_")[2].split(/\./).first
                if (diggit.to_i.to_s != diggit) then names_hash[text_file] = text_file.length end
                if (first_name =~ /\d/) then names_hash[text_file] = text_file.length end
                if (second_name =~ /\d/) then names_hash[text_file] = text_file.length end
        else
                names_hash[text_file] = text_file.length
        end
end

if second_folder != "err"
        Dir.glob(second_folder+"/*.*").each do |text_file|
```

```
                text_file = text_file.split("/").last
                if (text_file.split("_").length == 3) then
                        first_name = text_file.split("_")[0]
                        second_name = text_file.split("_")[1]
                        diggit = text_file.split("_")[2].split(/\./).first
                        if (diggit.to_i.to_s != diggit) then names_hash[text_file] = text_file.length
end
                        if (first_name =~ /\d/) then names_hash[text_file] = text_file.length end
                        if (second_name =~ /\d/) then names_hash[text_file] = text_file.length
end
                else
                        names_hash[text_file] = text_file.length
                end
        end
end

names_hash = Hash[names_hash.sort_by{|k,v| k} ]
names_hash = Hash[names_hash.sort_by{|k,v| v} ]

puts names_hash

CSV.open("results.csv","w") do |csv|
        names_hash.each do |element|
                csv << element
        end
end
```

Comments:

- File name is 'results.csv'.Otherwise the program works in most cases
- Code is readable if a bit too long
- This person seems to know what he was doing - a rare trait nowadays
- Details: Dir.glob traverses sub folders as well, digits in names deemed unacceptable

Quickfix: 56:CSV.open("results.csv","w") do |csv| -> CSV.open("result.csv","w") do |csv|

Rate: 5

Appendix 28

Author: Valentin Varbanov
Code:

```
=begin

Develop a program named FirstName_LastName_ClassNumber_041472.rb

1. you are given two arguments for a folders with files;
1.1 if there are other arguments they should be discarded;
2. file names in this folders are in the form First_Last_digits.rb;
3. find  the students that are only in the first folder and not in the second. A student is in both
folders if it there is a file with the same First and Last Name. Digits might be different;
4. Sort the result by Last name ;
5. Produce a result in CSV format named result.csv:

        LastName1,FirstName1
        LastName2,FirstName2
        ...
        LastNameN,FirstNameN


=end

students_first_dir = Array.new
students_second_dir = Array.new

for i in 0..1

        directory = ARGV[i]
        if ARGV[i].split(//).last(1).to_s == "/"
                directory += "**/*.rb"
        else
                directory += "/**/*.rb"
        end

        Dir.glob(directory).each do |dir|
                student = dir.split(/\//)
                if i == 0
                        students_first_dir.push(student)
                else
                        students_second_dir.push(student)
```

```
                end
        end
end

studentcsv = Array.new

students_first_dir.each do |std|
        match = 0
        students_second_dir.each do |std2|
                name = std.last.split(/_/)

                name2 = std2.last.split(/_/)
                for i in 0..1
                        if name[i] == name2[i]
                                match = 1
                        end
                end

        end
        studentcsv.push(name[1], name[2])
end

CSV.open("result.csv", "w") do |csv|
        studentcsv.each do |string|
                csv << string
        end
end
```

Comments:
Good idea about the file reading and name checks but lousy implementation:

- Separately checking the first and the last name - theres possibility of equal first or last names but not the combination of the two
- Name and name2 are local arrays they must be declared earlier or they'll be out of scope

Other problems:

- No require 'csv'
- String passed to csv method << instead of array or hash
- No last name sort
- Only writing first name to file

Quickfix: require 'csv'
name=[] name2=[] before inner loop
name check -> "#{name[0]} #{name[1]}" == "#{name2[0]} #{name2[1]}"
name sort -> studentscsv.sort_by{|fn,ln| ln}

Rate: 3

Appendix 29

Author: Veselina Kolova
Code:

```
=begin
Develop a program named FirstName_LastName_ClassNumber_65630e.rb

1. you are given an argument for a folder with files;
1.1 if there are other arguments they should be discarded
2. file names in this folder are in the form First_Last_digits.rb;
3. find all the students that have 5 letters in their second name;
4. Sort the result by First name DESC.
5. Produce a result in CSV format named result.csv:

                FirstName1,LastName1
                FirstName2,LastName2
                ...
                FirstNameN,LastNameN
=end

require 'csv'

people = Hash.new

Dir.glob("#{ARGV[0]}/**/*.*").each do |text_file|

        if File.extname(text_file) text_file.include?(".rb") &&
text_file.split(/_/).last.split(/\./).first.to_i.is_a Integer then
                if (text_file.split("/").last.split("_").length == 3) then
                        text_file = text_file.split("/").last
                        if (text_file.split("_")[1].length == 5) then
                                people[text_file.split("_")[1]] = text_file.split("_")[0]
                        end
                end
        end
end

people = Hash[people.sort_by{|k,v| k}.reverse]

CSV.open("result.csv","w") do |csv|
   people.each do |element|
   csv << element
```

```
   end
end
```

Comments:

- Not compiling - syntax errors(is_a?(not is_a),empty space in if instead of continuation)
- Long lines, unecessary if
- Hash with key last name will remove people with same surnames

Rate: 2

Appendix 30

Author: Vladimir Yordanov
Code:

```
#Develop a program named FirstName_LastName_ClassNumber_4bbed0.rb

#1. you are given an argument for a folder with files;
#1.1 if there are other arguments they should be discarded
#2. file names in this folder are in the form First_Last_digits.rb;
#3. find all the students that have 5 letters in their second name;
#4. Sort the result by Last Name ASC.

#5. Produce a result in CSV format named result.csv:

#               FirstName1,LastName1
#               FirstName2,LastName2
#               ...
#               FirstNameN,LastNameN



names = Hash.new
Dir.glob (ARGV[0] + "*.rb") do |file|
        if (ARGV[1] == true)
                ARGV[1] == false
        end

        slice = file.split("/").last
        first_name = slice.split('_')[0]
        second_name = slice.split('_')[1]
        if (second_name.length == 5)
                #print first_name
                #puts second_name
                names[first_name] = second_name
        end

end

names = names.sort
puts names

require 'csv'
```

```
CSV.open("results.csv", "w") do |csv|
        names.to_a.each do |element|
                csv << element
        end
end
```

Comments:

- Unnecessary dealing with ARGV[1]
- No format check
- Hash with key last name will remove people with same first names
- Writing in results.csv instead of result.csv
- Many variables
- Neat code

Rate: 3