E6

Instructions

November 9, 2015

You are going write three programs using a new programming language. Please follow the instructions below to confirm your eligibility, setup the virtual machine, learn the language, complete the writing tasks, complete surveys regarding your experience, and submit your data.

1 Eligibility

In the **past five years**, you have first 2.5 years' experience using imperative languages C/C++ (such as C/C++, Java, or Python).

2 Environmental setup

1. Launch the virtual machine by opening the file "LanguageFeaturesVM.vmwarevm" with VMware Fusion.
2. Adjust the VMware window to your screen, so that you can see both the lower-left corner with a trash-can icon and the upper-right corner with a gadget icon.
3. Launch the screen recorder by clicking the blue monitor icon on the left. A window titled "vokoscreen 1.9.0" should appear. [1]
4. Click "Start" in the vokoscreen window. The vokoscreen window should minimize automatically, and there should be a small white triangle to the left of the blue monitor icon that you have previously clicked.
5. Launch a terminal by clicking the black rectangle icon on the left. A terminal window titled "default@ubuntu:~\Desktop" should appear.

[1] For analyzing purposes, we record the screen in the virtual machine as you perform tasks in this experiment. We have configured vokoscreen to record the full screen, without audio, capturing 25 frames per second, encoding in the x264 format, storing into an avi file, and including mouse cursor movements. Feel free to double check these configurations and ask the experimenter if you have any questions.

3 Language tutorial

The language used in this experiment is similar to many other imperative languages such as C.

1. Learn about the language by reading the *Language Manual*.
2. **Double-click** the file "prog0.stu" on the desktop. An editor window titled "prog0.stu (~\Desktop) – gedit" should appear. This file contains an example program that counts the number of characters on each line of the file "input0.txt".
3. Run this example program by entering

   ```
   run prog0.stu
   ```

   in the terminal. You should see

   ```
   [Program return value]: 0.
   \5
   \3\0
   ```

   in the same terminal window. The first line presents the value returned from the program's `main` function. Then follows the outputs from `print` statements. The strings "\5", "\3", and "\0" indicate that there are 5, 3, and 0 characters on the first, the third, and the fourth lines, respectively. The newline character between "\5" and "\3" indicates that there are 10 characters on the second line, since the number 10 corresponds to the newline character `'\n'` in ASCII encoding.
4. Feel free to edit files "prog0.stu" and "input0.txt" and run the program again.

4 Writing tasks

There are three writing tasks. The first two tasks should help you complete the third task. Please complete them in order, and fill in the time that you start and finish each task.

4.1 Summing Matrix Elements

Started reading at 18 : 38 : 35

Your task is to fill in a program that sums up some elements in a 3×3 matrix, $\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}$. An input file specifies the elements to include. In this task and only in this task, you may assume that the input file is correct.

- **Input format:** Four ASCII-encoded digits:
  - i_0, the starting row index.
  - j_0, the starting column index.
  - h, the height.
  - w, the width.

  These numbers satisfy that $0 \le i_0 < i_0 + h \le 3$ and that $0 \le j_0 < j_0 + w \le 3$. Row and column indices start from 0.
- **Output format:** A byte that contains the sum of the elements between rows $i_0 \ldots (i_0 + h - 1)$ and columns $j_0 \ldots (j_0 + w - 1)$.

Figure 1 presents sample input and output. The program sums up all elements in the last two columns of the matrix: $1 + 2 + 4 + 5 + 7 + 8 = 27$. The `print` statement displays 27 as "\27".

```
0132
```

(a) Sample input

```
\27
```

(b) Sample output

Figure 1: Sample input and output

Finished reading at [illegible] : [illegible] : [illegible]

Please implement this program in file "prog1.stu", **save** the file, and run the program by entering "`run prog1.stu`" in the terminal. If there is no response, press Ctrl+C in the terminal and check if your program contains an infinite loop.

File "input1.txt" contains the sample input. Please test your program as you normally would.

Finished implementing at 18 : 47 : 27

After your implementation, please read the **sample solution** for this problem.

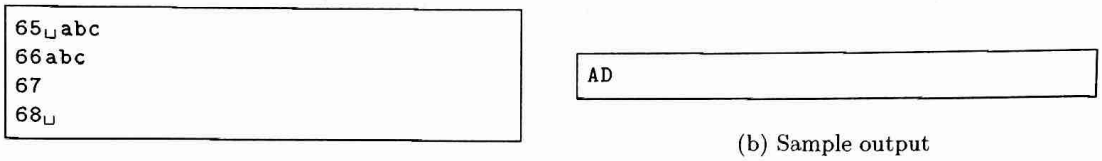
4.2 Converting ASCII Integers

Started reading at 18 : 49 : 37

Your task is to write a program that parses an integer preceding the first space character on each line of a file. Your program should be able to handle arbitrary inputs by skipping malformed lines.

- **Input format:** Each line starts with an ASCII-encoded, variable-length integer followed by a space character.
- **Output format:** Each output byte has the value of the leading integer from each input line.

Figure 2 presents sample input and output. Table 1 explains the sample output.



(a) Sample input

(b) Sample output

Figure 2: Sample input and output

Table 1: Explanation for sample output

Input line	Output byte	Explanation
1	65	"`print(65);`" displays `'A'`
2	(skipped)	unexpected character `'a'`
3	(skipped)	the trailing space character is missing
4	68	"`print(68);`" displays `'D'`

Finished reading at 18 : 52 : 26

Please implement this program in file "prog2.stu", **save** the file, and run the program by entering "`run prog2.stu`" in the terminal. If there is no response, press Ctrl+C in the terminal and check if your program contains an infinite loop.

File "input2.txt" contains the sample input. Please test your program as you normally would.

Finished implementing at 19 : 04 : 31

After your implementation, please read the **sample solution** for this problem.

4.3 Image Thumbnail Generator

Started reading at 19 : 06 : 06

Your task is to write a program that generates thumbnails for multiple bitmap images in a file. Your program should be able to handle arbitrary inputs by skipping malformed images.

Image thumbnails are small images that capture the overall shape of large images. For example, a thumbnail for Figure 3 is shown in Figure 4.

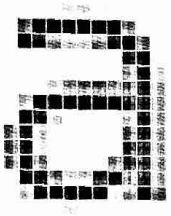


Figure 3: Magnified original image

Figure 4: Magnified thumbnail

You are going to use the following algorithm that averages neighboring pixel values. Given a scaling factor s, the thumbnail for an image of height h and width w has height $\lfloor h/s \rfloor$ and width $\lfloor w/s \rfloor$. The value of the pixel in row i and column j of the thumbnail is the floor average of the values of all pixels in the s^2 square area between rows $i \cdot s \ldots (i \cdot s + s - 1)$ and columns $j \cdot s \ldots (j \cdot s + s - 1)$ of the original image.

- **Input format:** Each line describes an original image. Each line contains the following contents separated by a single space character:
  - An image name, which is a string of 1–10 characters long.
  - An ASCII-encoded variable-length integer, s, that represents the scaling factor.
  - An ASCII-encoded variable-length integer, h, that represents the original image height.
  - An ASCII-encoded variable-length integer, w, that represents the original image width.
  - $h \cdot w$ consecutive digits that each represents a pixel value. The pixels are ordered as follows: inside each row, pixels are ordered from left to right; the rows are each grouped together and ordered from top to bottom. Each pixel has one of ten possible values: '0'...'9'.
- **Output format:** The output contains lines that each describes a thumbnail. Each line contains the following contents separated by a single space character:
  - The image name.
  - $\lfloor h/s \rfloor \cdot \lfloor w/s \rfloor$ consecutive digits that each represents a pixel value.

Figure 5 presents sample input and output. Table 2 explains the sample output.

```
Img1␣2␣2␣2␣1234
Img2␣2␣4␣4␣1234567890123456
Img3␣2␣1␣2␣12
Img4␣3␣3␣4␣123456789012
```

(a) Sample input

```
Img1␣2
Img2␣3543
Img3␣
Img4␣3
```

(b) Sample output

Figure 5: Sample input and output

Table 2: Explanation for sample output

Name	Original	Thumbnail	Explanation
Img1	1 2 3 4	2	With $s = 2$, the thumbnail for Img1 has height $\lfloor 2/2 \rfloor = 1$ and width $\lfloor 2/2 \rfloor = 1$. The single pixel has the value of $\lfloor (1+2+3+4)/4 \rfloor = \lfloor 10/4 \rfloor = 2$.
Img2	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6	3 5 4 3	With $s = 2$, the thumbnail for Img2 has height $\lfloor 4/2 \rfloor = 2$ and width $\lfloor 4/2 \rfloor = 2$. The top-left pixel in the thumbnail has the value of $\lfloor (1+2+5+6)/4 \rfloor = 3$. The top-right pixel in the thumbnail has the value of $\lfloor (3+4+7+8)/4 \rfloor = 5$. The bottom-left pixel in the thumbnail has the value of $\lfloor (9+0+3+4)/4 \rfloor = 4$. The bottom-right pixel in the thumbnail has the value of $\lfloor (1+2+5+6)/4 \rfloor = 3$.
Img3	1 2	(empty)	With $s = 2$, the thumbnail for Img3 has height $\lfloor 1/2 \rfloor = 0$. This height indicates that the thumbnail image is empty.
Img4	1 2 3 4 5 6 7 8 9 0 1 2	3	With $s = 3$, the thumbnail for Img4 has height $\lfloor 3/3 \rfloor = 1$ and width $\lfloor 4/3 \rfloor = 1$. The single pixel has the value of $\lfloor (1+2+3+5+6+7+9+0+1)/9 \rfloor = \lfloor 34/9 \rfloor = 3$.

Finished reading at 19 : 14 : 14 .

Please implement this program in file "prog3.stu", **save** the file, and run the program by entering "`run prog3.stu`" in the terminal. If there is no response, press Ctrl+C in the terminal and check if your program contains an infinite loop.

File "input3.txt" contains the sample input. Please test your program as you normally would.

Finished implementing at 19 : 50 : 00 .

5 Survey

5.1 Summing Matrix Elements

Mental Demand. How mentally demanding was task 1?
Please rate from 0 (very low) to 10 (very high).

5

Temporal Demand. How hurried or rushed was the pace of task 1?
Please rate from 0 (very low) to 10 (very high).

5

Performance. How successful were you in accomplishing task 1?
Please rate from 0 (failure) to 10 (perfect).

8

Frustration. How insecure, discouraged, irritated, stressed, and annoyed were you?
Please rate from 0 (very low) to 10 (very high).

3

How many times did you cheat? Where did you cheat?
Don't worry, there is no punishment for cheating.

0

Any other comments or clarification?

5.2 Converting ASCII Integers

Mental Demand. How mentally demanding was task 2?
Please rate from 0 (very low) to 10 (very high).

6

Temporal Demand. How hurried or rushed was the pace of task 2?
Please rate from 0 (very low) to 10 (very high).

5

Performance. How successful were you in accomplishing task 2?
Please rate from 0 (failure) to 10 (perfect).

8

Frustration. How insecure, discouraged, irritated, stressed, and annoyed were you?
Please rate from 0 (very low) to 10 (very high).

6

How many times did you cheat? Where did you cheat?
Don't worry, there is no punishment for cheating.

0

Any other comments or clarification?

5.3 Image Thumbnail Generator

Mental Demand. How mentally demanding was task 3?
Please rate from 0 (very low) to 10 (very high).

8

Temporal Demand. How hurried or rushed was the pace of task 3?
Please rate from 0 (very low) to 10 (very high).

8

Performance. How successful were you in accomplishing task 3?
Please rate from 0 (failure) to 10 (perfect).

6 :(

Frustration. How insecure, discouraged, irritated, stressed, and annoyed were you?
Please rate from 0 (very low) to 10 (very high).

9

Any other comments or clarification?

If possible, please add more features, say, it can take more than 1 global var or func can take more than one input ...

But overall it's fine. It's a bit stressful but fun :).

6 Submission

1. Stop the screen recorder by clicking the blue monitor icon on the left and then clicking "Stop" in the window named "vokoscreen 1.9.0".
2. Shut down the virtual machine by clicking the gadget icon on the top-right corner, the "Shut Down..." option at the bottom of the drop-down menu, and the "Shut Down" icon on the right. DO NOT shut down from the VMware menu.
3. Send your updated file "LanguageFeaturesVM.vmwarevm" back to the experimenter.
4. Turn in this document.