Lab 05 Specification – Expressions in Racket and Postscript Languages
50 points

(Individual Lab -) Due by: 10/04/2021 2:00 PM

Lab Goals

- To learn to use prefix and postfix notations for writing expressions.
- To obtain a basic understanding of Scheme/Racket and PostScript programming languages.
- To reflect on the different types of expression notations and outline advantages and disadvantages associated with all.

Learning Assignment

If not done previously, it is strongly recommended to read all of the relevant "GitHub Guides", available at the following website:
`https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading assignment is useful to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, it is also recommended to do the reading assignment from the section of the course textbook outlined below:

- PLP chapter 06, section [6.1 - 6.4]

Assignment Details

In this lab, students will practice understanding and developing program(s) in new languages and experiment with developing control flow in expressions. This is a good practice to retain the knowledge from the class discussions so far.

Students are not expected to achieve technical mastery in any single language in this course. Instead, we are trying to learn different languages in the context of studying the principles of programming languages. In other words, our goal is to put the principles of programming languages at the forefront and not achieving language mastery.

At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL(s) to clarify if there is any confusion related to the lab and/or class materials. The Professor proofread the document more than once, if there is an error in the document, it will be much appreciated if you can communicate that to the Professor. The class will be then informed as soon as possible regarding the error in the document. Additionally, it is highly recommended that students will reach out to the Professor in advance of the lab submission with any questions. Waiting till the last minute will minimize the student's chances to get proper assistance from the Professor and the Technical Leader(s).

Students are recommended to get started with this part in the laboratory session, by discussing ideas and clarifying with the Professor and the Technical Leader(s). Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.

2. Submitting a copy of the other's program(s) and technical reports is strictly not allowed. Please note that all work done during lab sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work, students maximize the learning and increase the chances to do well in other assessments such as skill tests, exams, etc · · ·

Preliminary Steps

It is important that you can set up Docker and GitHub to complete the rest of the lab. Please follow the guidelines below to complete the preliminary steps.

1. **[Docker Setup.]** At this point, I expect the MAC, Linux, and Windows Pro users, to have this step completed based on our previous class discussions. For those who had not completed this step, the documentation below should provide more details regarding the download and installation setup.

   - Get Docker setup completed on your laptops:
   - Docker Mac Setup:

     `https:/docs.docker.com/docker-for-mac/install/`
   - Docker Ubuntu Setup

     `https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04`
   - Docker Windows Setup:

     `https:/docs.docker.com/docker-for-windows/install/`
   - If the setup goes correctly as desired, you should be able to get started and validate the Docker version and run the hello world docker container using the following commands:

     docker –version

     docker run hello-world
   - There are some more documentation for Docker get started to test your installation in the link provided below:

     `https:/docs.docker.com/docker-for-mac/`

     `https:/docs.docker.com/docker-for-windows/`

2. **[Loading Docker Container.]** There are two steps in loading the container, namely:

   - Build the container
   - Connect and Run the container

   Build the container: So to build the container. the following steps should be performed.

   (a) First, accept the lab URL provided in Slack. After downloading the lab folder from the GitHub classroom, navigate to the cmpsc201-fall-21-lab05 directory using terminal (Mac/Ubuntu) or Command Prompt/Docker quick start terminal (windows).

   (b) Build the docker image using the following command:

   docker build -t cs201lab05 .

   Please note, you are required to have the period in the command above.

   (c) Note: In the command above, cs201lab05 is the user-provided image name. This could be random. But it is recommended to use the same name to easily follow the rest of this document. Additionally, it is required to be inside the `cmpsc201-fall-21-lab05` directory to run the build command. If you are not inside the `cmpsc201-fall-21-lab05` directory, you may receive an error message.

   (d) Upon successful build, it is recommended to verify the correctness of image creation by using the following command:

   docker image ls

   (e) The image named "cs201lab05" should be listed as one of the outputs from the command above.

   Connect and Run the container: So to create and run the container. the following steps should be performed.

   (a) Run the docker container based on the image created in the previous steps using the following command:

   Mac/Ubuntu:

   ```
   docker run --rm -v $(pwd)/src:/root -it cs201lab05
   ```

   Windows:

   ```
   docker run --rm -v "%cd%/src":/root -it cs201lab05
   ```

   Ignore if you receive a bash command not found error while connecting to the docker container. The command above simply takes us into the docker container so that we can start to execute the code.

   (b) If you get an error while executing the command above on your windows laptop, then type the following:

   ```
   docker run --rm -v $pwd\src:/root -it cs201lab05
   ```

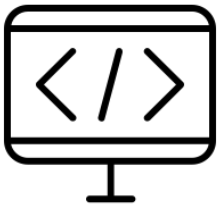
   (c) To run the above command, it is required to be inside the `cmpsc201-fall-21-lab05` directory. And, please note, you will log in to the container after entering the above command.

   (d) After creating the container, the run command above creates a mount between the host machine and the container with a shared folder space. So, any files placed inside the host mount directory can be easily accessible inside the container mount directory and vice versa. For Windows laptop, you may need to install Notepad++ and convert **CRLF to LF** as we did in lab2 for all the files in src directory if you get an error.

3. **[GitHub Setup.]** Take a look at the detailed documentation for getting started with GitHub, which is available at: `https://docs.github.com/en/get-started`

   You are required to know the procedure to git clone, git pull, git add, git commit, and git push to access the lab specification folder and to submit your lab for grading purposes. If there is an issue with your GitHub setup please discuss it with your Technical Leader(s) and/or the Professor.

Part 01 - Evaluate prefix expressions using Racket (25 points)



Racket is a dialect of Scheme, which in turn is an extension of a very old language called LISP. We have an environment devoted entirely to Racket, called "DrRacket". You can access it from the docker container by executing the hello racket program. In the terminal, after connecting to the container just type `racket hello.rkt`. This would display the hello world message on the console. This racket program file implement one or more LISP commands that is executed while running the racket file.

Study the tutorial provided on `https://docs.racket-lang.org/quick/` and try the examples discussed in the tutorial. You may also use the complete documentation as a reference: `https://docs.racket-lang.org/guide/`. Take a look at the examples in this section, for some sample racket code written by the Professor:

1. The racket program named **area-sphere.rkt** program file contains an implementation to calculate the area of Sphere. This program file can be executed inside the container by just typing `racket area-sphere.rkt`
2. The racket program named **area-triangle.rkt** contains an implementation to calculate the area of Triangle. This program file can be executed inside the container by just typing `racket area-triangle.rkt`
3. The racket program named **strlen.rkt** contains an implementation to calculate the length of the given string. This program file can be executed inside the container by just typing `racket strlen.rkt`

After you have studied some Racket code, develop a new Racket file and implement the tasks outlined below. Make sure to **cd** into the **part-1** directory to access the racket files. These racket source code files may be opened from any text editor of your choice.

1. The racket program named **volume-cone.rkt** is empty. Define two symbols `r` with value 10 and `h` with value 5 resp, then write a one-line expression describing the volume of a right circular cone of radius `r` and height `h`. The formula to calculate volume of the cone is:

   $3.14 \times r^2 \times \frac{h}{3}$

2. The racket program named **volume-sphere.rkt** is empty. Define a new symbol `r` with value 10, then write a one-line expression describing the volume of a sphere of radius `r`. Solve fractions and exponents by implementing your own expressions by referring the **area-sphere.rkt** program file. The formula to calculate volume of the sphere is:

   $\frac{4}{3} \times 3.14 \times r^3$

3. The racket program named **expressions.rkt** is empty. Define symbols for `a`, `b`, `c`, and `x` with values 1.2, 2.3, 3.4, and -2, respectively. Write a one-line expression that finds the value of $ax^2 + bx + c$.

4. The racket program named **midpoint.rkt** is empty. Define a symbol `s` with value `"Hello, Racket"`. Write a one-line expression that defines a symbol `mid` equal to half the length of `s`, rounded down to the nearest integer. **Round** is a builtin function to round down to the nearest integer. Refer the `string-length` function used in **strlen.rkt** program file that is used to compute the length of the string. Calculate this in your expression, don't just enter a constant! Display the value of `mid` (i.e., just write the expression `mid`).

5. The racket program named **stringify.rkt** is empty. Define a symbol `s` with value `"Hello, Racket"`. Write a one-line expression that defines the symbols `s` and `mid` similar to the one that you defined previously. Write a one-line expression using `substring`, `string-append`, and the constant string `"Dr."` that creates the string `"Hello, Dr. Racket"`. Feel free to look up online for technical details related to the use of these built in functions.

Make sure you have commented all portions of your program and save your Racket files in your repository.

Part 02 - Evaluate some postfix expressions using Postscript (15 points)

Postscript is a file format created to make it easy for computers to create vector images. To help computers quickly define complex operations, it also included a computer programming language which could be used to move or repeat objects without having to redefine them. Postscript primarily use the **postfix notation** to implement expressions. A postfix expression is a collection of operators and operands in which the operator is placed after the operands. That means, in a postfix expression the operator follows the operands. The lab repository has a few sample PostScript files. Refer the ex1.ps, ex2.ps, and ex3.ps files in the examples directory.

To visualize the **output** of the Postscript files, just run the command in the file named **pdf-converter-command** inside your docker container. Make sure to cd into the **examples** directory. This command converts all the .ps files into a PDF file. Once this process is complete, then the PDF file can be viewed using any PDF viewer. It is very important to make sure that the command is executed inside the Docker container and you are inside the directory that contains all the `.ps` files. Make sure to run this command, every time a new .ps file is created so that the PDF file can be used to visualize the Postscript and for testing purposes.

Study the code of these file (by opening them with a text editor). There are many online resources with Postscript examples and you may find a complete PostScript reference on:
`https://www.adobe.com/jp/print/postscript/pdfs/PLRM.pdf`.

The commands that are helpful to develop Postscript files are provided below:

- *x y* `moveto` – move the pen to location (x, y)
- *rx ry* `rmoveto` – move the pen rx in the x-direction, ry in the y-direction. In other words, move *relative to* the current position. If the pen is at $(100, 200)$, then "`20 30 rmoveto`" will place it in location $(120, 230)$.
- *x y* `lineto` – draw a line to location (x, y)
- *rx ry* `rlineto` – draw a line rx in the x-direction, ry in the y-direction (in other words, draw *relative to* the current location)
- *x y* `add` – same as $x + y$
- *x y* `sub` – same as $x - y$
- `/x` *expression* `def` – assign $x = expression$
- `stroke` – When you complete a figure, this "strokes the ink" onto the page; it does not change the location of the pen
- `showpage` – should be the last thing in your file.

Develop two PostScript files that use no other Postscript Commands besides the ones described previously. Make sure to **cd** into the **part-2** directory. The name of the first file is `pscript1.ps`. In this file, you should draw at least four different figures (not rectangles or triangles) in four different places on the page. Comments in your program should describe these (e.g., "%Draws an 'F' shaped polygon in the upper right corner"); further comments are welcome, of course.

The name of the second file is `pscript2.ps`. The second file should be a simple composition of your own devising. It should use simple arithmetic expressions. It should have at least four distinct components (which could be shapes or lines). In your comments you may make an artistic statement about your composition. One such composition, is provided in ex3.ps file. Make sure to execute the command in the `pdf-converter-command` to convert the .ps files into a new PDF file named **pscripts.pdf**. Note: It is required to be in the **part-2** directory for running this command.

Part 03 - Technical Report (10 Points)

Conduct an online research to answer the questions provided in this section. In the `Technical-Report` file, add your technical points related to the questions below:

- List out your view of using Prefix Notations, and discuss advantages and disadvantages.
- List out your view of using Postfix Notations, and discuss advantages and disadvantages.
- List out your view of using Infix Notations, and discuss advantages and disadvantages. First research about Infix Notations and identify which language that you used previously used Infix Notations.
- Cite your references at the bottom of the report.

Part 04 - Honor Code

Make sure to **Sign** the following statement in the `honor-code.txt` file in your repository. To sign your name, simply replace Student Name with your name. The lab work will not be graded unless the honor code file is signed by you.

This work is mine unless otherwise cited - Student Name

Part 05 - Reflection

Add a Reflection to the repository by modifying the `reflection` file in the lab repository. List out the biggest learning points and any challenges that you have encountered during this lab.

Submission Details

For this assignment, please submit the following to your GitHub repository by using the link shared to you by the Professor:

1. Modified source code in `volume-cone.rkt` file.
2. Modified source code in `volume-sphere.rkt` file.
3. Modified source code in `expressions.rkt` file.
4. Modified source code in `midpoint.rkt` file.
5. Modified source code in `stringify.rkt` file.
6. Modified source code in `pscript1.ps` and `pscript2.ps` files.
7. Output PDF file named `pscripts.pdf` file.
8. A document containing the technical report in the file named `Technical-Report` that provides the answers to the questions asked.
9. A document containing the reflection of the lab in the file named `reflection`.
10. A signed honor code file, named `Honorcode`.
11. To reiterate, it is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus.

Grading Rubric

1. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits may be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your GitHub repository will lead to receiving no points given for the lab submission. In this case, there is no solid base to grade the work.
3. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before the due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in GitHub. In this way, an updated version of the student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then, in this case, there are no points awarded to the student.
4. If a student needs any clarification on their lab grade, it is strongly recommended to talk to the Professor. The lab grade may be changed if deemed appropriate.