*Ho Chi Minh University of Technology*

*Faculty of Computer Science an Engineering - Computer Science Major*

*Course: Discrete Structure for Computing*

--------------------o0o--------------------



**HK212**

**Assignment Report  
Co- Lecturer**: Nguyễn Tiến Thịnh - Nguyễn Văn Minh Mẫn

**Class**: CC01 - **Group**: 3

|  |
| --- |
| Team member |

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **Student ID** |
| 1 | Vũ Châu Duy Quang | 2153730 |
| 2 | La Cẩm Huy | 2153379 |
| 3 | Đỗ Lâm Ngọc Thức | 2153143 |
| 4 | Nguyễn Vĩnh Huy | 2152597 |

**TABLE OF CONTENT**

[I. Theories Recall 3](#_Toc11818)

[1. Mathematical Theories: 3](#_Toc19283)

[ Prefix notation: 3](#_Toc29536)

[ Postfix notation: 3](#_Toc1998)

[ Infix notation: 3](#_Toc17206)

[2. Computing Theories: 3](#_Toc2969)

[ Graph theory: 3](#_Toc21644)

[ Tree - Binary Tree: 3](#_Toc7566)

# Theories Recall

## Mathematical Theories:

### Prefix notation:

A parenthesis-free notation for forming mathematical expressions in which each operator precedes operands.

Ex: 6+5 can be written in Prefix notation: + 6 5

### Postfix notation:

A parenthesis-free notation for forming mathematical expressions in which each operator follows its operands.

Ex: 7\*2 can be written in Post fix notation: 7 2 \*

* **Infix notation:**

A parenthesis-free notation for forming mathematical expressions in which each operator is in between its operands. This is also know as the way we normally use to express functions or calculations.

## Computing Theories:

* **Graph theory:**

Graph theory is the study of graphs, which are mathematical structures used to model

pairwise relations between objects.

* **Tree - Binary Tree:**

Tree structure or a tree is an undirected graph in which any two vertices are connected by exactly one path

A binary tree is know as a tree data structure in which each node has at most two children,

which are often called as the left child and the right child.

# **Idea and execution**

1. **Problems:**

P1: Write a function receive a constant string which is:

1. An infix arithmetic notation expression. Return another string which is it’s prefix notation
2. An infix arithmetic notation expression. Return another string which is it’s postfix notation
3. A prefix or postfix arithmetic notation expression. Return the value of the expression.

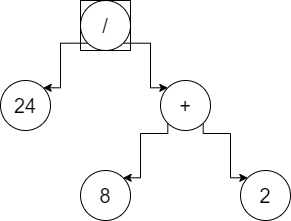
P1: Write a function receive a constant string which is:

1. An infix logical notation expression. Return another string which is it’s prefix notation
2. An infix logical notation expression. Return another string which is it’s postfix notation
3. A prefix or postfix logical notation expression. Return the value of the expression.
4. **Idea:**

The main idea to solve the problems listed above is to use a binary tree structure to store the expression. Then use the algorithm provided in the problem (PreOrder Tree Traversal, PostOrder Tree Traversal and Postfix Evaluation) to transform or evaluate the expression given which is converted to binary tree above.

Recall that an operator only have 1 or 2 operands (this is true for both arithmetic and logical expression) which is perfectly fit with the binary tree definition (The operator will be parents and its operands or smaller expression will be the child node).

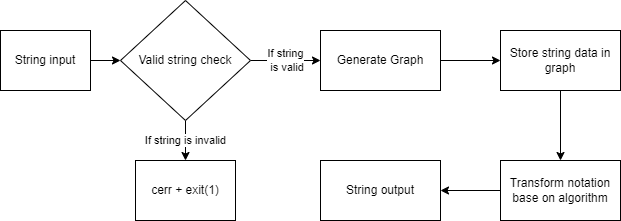
Ex: 24/(8+2) can be store in a binary tree like this:



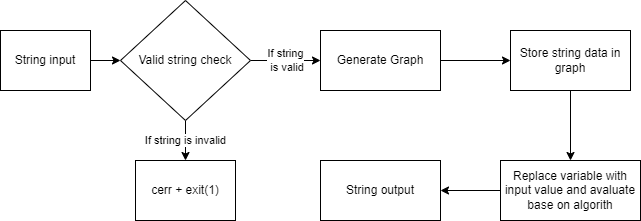
1. **Execution:**

Knowing the main idea to solve the problems, we come up with a simple yet efficient way to handle it.

Flow chart for transform infix notation to others notation: (P1a, 1b,2a, 2b)



Flow chart for calculate the value of notation:



\*\*Note: Since logical variable only have 2 type of input value (either 1 for true and 0 for false), we must build a function in other to check for valid input of these variable in order to evaluate.

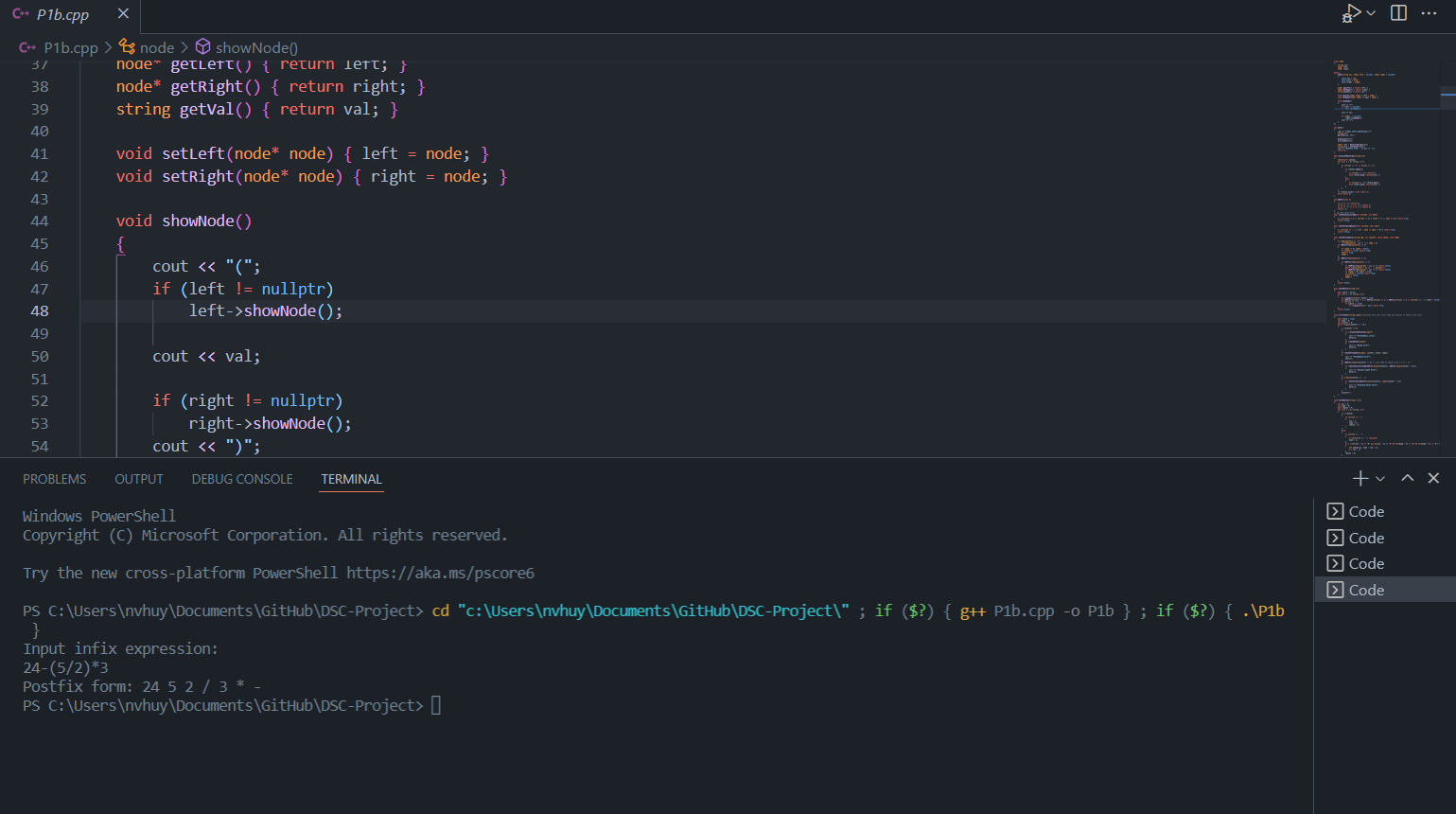
Looking at the flowcharts, it can be seen that building function base on usage would cost less time, especially function that can be reused in all of the 6 problems , such as graph generate function or valid input check function. So we decided to start working that way in order to save time and reduce the amount of work we need to do.

Also, with the current COVID-19 situation in Sai Gon, we decided that it would be best to work and communicate over GitHub instead of direct meeting, and it turn out to be a great idea.

# Result

## P1a

## P1b



## P1c

## P2a

## P2b

## P2c

# **Conclusion**

Finalizing the solution, we have a look back in the process and see that our work have some advantage and disadvantage we want to share about

## **Advantage:**

* Building function base on usage and reusability, which can reduce work and time spent
* Remote working and communication via GitHub
* Decent use of graph and tree, especially binary tree in solution

## **Disadvantage**

* Program is not fully debugged due to lack of time and tight schedules (We need to work for other assignments and for final exam as well)
* Lack direct communication (most of our communication is via facebook and GitHub), but we think that this work just fine)

## **Final conclusion**

Although tree and graph are strong data structure and can be use to solve complex problems, it is indeed hard and time-consuming trying to fully understand and apply these structure in real life programming, especially with the lack of time. However, we been working and researching about the problem for a while, and we think that we have a decent work and gain a great amount knowledge about this.