Programming Languages 2020 Spring Project 2

Due: 16 June 2020

Groups: Groups should consist of 2 members at most. (1 or 2 people)

Task: Path_maker is a basic scripting language for creating directory trees. (Rules of the language are below) Write a basic interpreter for Path_maker

Path_maker Language Description

Data Types: Path is the only data type. Path constants are relative directory path expressions written in the form:

<dir1/dir2/dir3> where dir1, dir2 and dir3 are directory names

- No file names are of any concern (just directories)
- Directory names start with a letter (upper or lower case) and are made of any combination of letters, digits and underscore characters (only). (Punctuation characters are not allowed. Blank characters are not allowed either.) Directory names are not case sensitive so <AA> and <aa> are basically the same. (since that is the policy of most operating systems)
- Operator "*" can be used instead of a directory name, and it indicates parent directory. It can be used multiple times before any other directory name.

  Example: <*/*/mydirectory> indicates that one should move up (to parent) twice and then choose mydirectory.
- Operator * can only be used at the beginning of path expressions. <hi/*/there> is not allowed.
- Operator / cannot be used at the beginning or the end of any path. So </hi/there> is not allowed. Neither is <hi/there/> allowed.
- Blanks in a path expression are ignored (unless they exist in a directory name (which is not allowed)) so < * /* / mydirectory> is OK.

Variables: There are no variables in the language.

Basic Commands: The only two basic commands are “make” and “go”. Make has the form:

make <myDirectoryPath>;

It simply creates the directories in the myDirectoryPath. If the path already exists it does nothing (but gives a warning message). If the path partially exists, it completes the path.

Example: make <*/project1/data> goes up once and then creates a directory called project1 and then creates another one called “data” inside it.

“make” does not change the current (working) directory. (This is what we do with go command)

“go” simply changes the current directory

Syntax: go <myPathExpression>;

If the path does not exist, go does nothing. (Gives an error message but does not exit the execusion) It does not partially follow a path. Partial existence of any path is considered as inexistence.

Control Structures: There is an “if” clause and a similar “ifnot” clause

if <path_expression> **command**

is the basic form of this clause where command can be a basic command or a **block.** “if” clause executes the **command** if from the current directory the path <path_expression> exists.

“if” does not change the current directory.

“ifnot” clause has the exact same structure

ifnot <path_expression> **command**

but operates if the path <path_expression> does **not** exist.

Blocks: A command can be a basic command (“make” or “go”) but it can also be a block. A block is a list of lines of code enclosed in { } set brackets. Blocks may also be nested in one another.

End of line character: Only “make” and “go” commands require an end of line character and it is ‘;’ (semi-colon)

Keywords: Keywords are case sensitive and all are lowercase. They are:

make, go, if, ifnot

Symbols: < , > , { , } , / , *

Code Example:

```
if <*>
{  go <*>;  make <data/doctors>;
   if <user/ahmet> go <path_expression>;
}
```

Project Details

Input: Prompt for the source file name. Get the source file name from the user. Assuming the user gave the name x, you should open and read the source file with the extension ".pmk" like x. pmk

Output: Your program should run the Path_maker script and create the necessary directory tree. It should catch the errors and create intelligible error messages (that can help the people correct their mistakes).

Language: C (strictly)

Submission: You should submit a report explaining the analysis phase which should include a detailed description of how you parse the pmk files. Your report should also include the printout of screenshots and of course the source code. You should submit your work in a zip file containing a soft copy of your report and source code of your program to EgeDers systems under "Project 2".

Late submission: Recursively %10 punishment for each extra day.