Project for CPSC323

Fall 2025

The programming assignments are based on a language called "Rat25F" which is described as follows. The Rat25F language is designed to be an easy to understand. It has a short grammar and relatively clean semantics.

You will need to form a group of 3 (one personal should be responsible for one assignment) for the duration of the entire project (i.e., all three assignments).

- **Submit the group sign up sheet by 9/14 (11:59 pm)**

RAT25F

1) Lexical Conventions:

The lexical units of a program are identifiers, keywords, integers, reals, operators and other separators. Blanks, tabs and newlines (collectively, "white space") as described below are ignored except as they serve to separate tokens.

Some white space is required to separate otherwise adjacent identifiers, keywords, reals and integers.

<Identifier> is a sequence of letters, digits, or "$". However, the first character must be a letter. **Upper and lower cases are same.**

<Integer> is an unsigned decimal integer i.e., a sequence of decimal digits.

<Real> is an optional integer followed by "." and Integer, e.g., 123.00 or .001 but not 123.

Some identifiers are reserved for use as ***keywords***, and may not be used otherwise:

e.g., integer, if, else, fi, while, return, get, put etc.

Comments are enclosed in **"** **"** and should be **entirely ignored** by the LA, SA etc.

2) Syntax rules : The following BNF describes the Rat25F.

R1. <Rat25F> ::= <Opt Function Definitions> # <Opt Declaration List> <Statement List> #
R2. <Opt Function Definitions> ::= <Function Definitions> | <Empty>
R3. <Function Definitions> ::= <Function> | <Function> <Function Definitions>
R4. <Function> ::= function <Identifier> (<Opt Parameter List>) <Opt Declaration List> <Body>
R5. <Opt Parameter List> ::= <Parameter List> | <Empty>
R6. <Parameter List> ::= <Parameter> | <Parameter> , <Parameter List>
R7. <Parameter> ::= <IDs > <Qualifier>
R8. <Qualifier> ::= integer | boolean | real
R9. <Body> ::= { < Statement List> }
R10. <Opt Declaration List> ::= <Declaration List> | <Empty>
R11. <Declaration List> := <Declaration> ; | <Declaration> ; <Declaration List>
R12. <Declaration> ::= <Qualifier > <IDs>
R13. <IDs> ::= <Identifier> | <Identifier>, <IDs>
R14. <Statement List> ::= <Statement> | <Statement> <Statement List>
R15. <Statement> ::= <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan> | <While>
R16. <Compound> ::= { <Statement List> }
R17. <Assign> ::= <Identifier> = <Expression> ;
R18. <If> ::= if (<Condition>) <Statement> fi |
if (<Condition>) <Statement> else <Statement> fi
R19. <Return> ::= return ; | return <Expression> ;
R21. <Print> ::= put (<Expression>);

R21. <Scan> ::= get (<IDs>);
R22. <While> ::= while (<Condition>) <Statement>
R23. <Condition> ::= <Expression> <Relop> <Expression>
R24. <Relop> ::= == | != | > | < | <= | =>
R25. <Expression> ::= <Expression> + <Term> | <Expression> - <Term> | <Term>
R26. <Term> ::= <Term> * <Factor> | <Term> / <Factor> | <Factor>
R27. <Factor> ::= - <Primary> | <Primary>
R28. <Primary> ::= <Identifier> | <Integer> | <Identifier> (<IDs>) | (<Expression>) | <Real> | true | false
R29. <Empty> ::= ε

Note: <Identifier>, <Integer>, <Real> are token types as defined in section (1) above

3) Some Semantics

- Rat25F is a conventional imperative programming language. A Rat25F program consists of a sequence of functions followed by the "main body" where the program executes.
- **All variables and functions must be declared before use.**
- Function arguments are passed by value.
- There is an implied expressionless return at the end of all functions; the value returned by expressionless return statement is undefined.
- Arithmetic expressions have their conventional meanings.
- Integer division ignores any remainder.
- Type casting is not allowed (e.g., assigning an integer to a real variable)
- No arithmetic operations are allowed with booleans (e.g., true + false)
- Others, as we will define during the semester

4) A sample Rat25FProgram

```
" this is comment for this sample code which
  converts  Fahrenheit into Celcius "

function convert (fahr integer)
{
        return 5 * (fahr -32) / 9;
}

#
    integer    low,  high, step;      "declarations "

    get (low, high, step);
    while (low  <=  high )
       {  put (low);
          put (convert (low));
          low =  low  +  step;
       }

 #
```

VERY, VERY IMPORTANT !!!

For each programming assignment, you should turn in the following
(one submission per group)

1) Softcopy of each assignment:

a) Cover page
b) About 2 pages of documentation (see the Documentation template)
c) Source code listing with proper comments for each procedure, sections if necessary.
d) Test cases (input files). Find at least **3 test cases (< 10 , < 21, > 21 source lines)**
e) **The results (output files) of the test cases**
f) Executable file (e.g., .exe, .jar) under "Window" or "Unix/Linux" OS

Use the "submit" feature on Canvas
Before submission, put all files into a directory, zip the directory and submit one "zipped" file.

Note: 1. I must be able to run your program in order to give you a grade.

2. If you turn in a program that **cannot be run**, there will be an automatic 2 points deduction.

3. If you **don't turn in the documentation**, there will be an automatic 2 points deduction.

4. I will accept late project however, there will be some deductions:
2 points deduction for the first day and 0.1 each day you are late (from max of 10). For example, if you are late for one week, then the maximum point you will get is $10 - (2 + 6 * 0.1) = 7.4$

Final Notes:
- You will most likely not pass this course without doing the projects.
- The projects are built on each other,
so make sure that you do well the first project