

Project for CPSC323

Spring 2018

The programming assignments are based on a language called "Rat18S" which is described as follows. The Rat18S 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 2 or 3 for the duration of the entire project (i.e., all three assignments).

RAT18S

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 or digits, however, the first character must be a letter and last char must be either \$ or letter. Upper and lower cases are same.

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

<**Real**> is integer followed by “.” and Integer, e.g., 123.00

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

e.g., int, if, else, endif, while, return, get, put etc.

Comments are enclosed in ! !

2) Syntax rules : The following BNF describes the Rat18S.

```
R1. <Rat18S> ::= <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> ::= int | 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> endif |
               if ( <Condition> ) <Statement> else <Statement> endif
R19. <Return> ::= return ; | return <Expression> ;
```

```

R20. <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> ::= ■

```

3) Some Semantics

- Rat18S is a conventional imperative programming language. A Rat18S 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 Rat18S Program

! this is comment for this sample code which
converts Fahrenheit into Celcius !

```

function convert$ [fahr:int]
{
    return 5 * (fahr -32) / 9;
}

%%%
int 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:

- 1) **A hardcopy of**
 - a) Cover page
 - b) About 2 pages of documentation (see the Documentation template)
 - c) Source code listing with proper comments for each procedures, sections if necessary.
 - d) Test cases (input files). Find at least **3 test cases (< 10 , < 20, > 20 source lines)**
 - e) The results (output files) of the test cases
- 2) **You will also to submit a softcopy of all mentioned in (1) and executable file (e.g., .exe) using the “submit” feature on Titanium.**

Before submission, zip all of your files and submit one “zipped” file

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

Run your program using any computer in CS building before turning in.

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 $10 - (2 + .6) = 7.4$

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