Project: Compiler and Virtual Machine for A Programming Language.

1. **Sample Program written in BRU:**

```
// programs written must be saved with '.bru' extension

method factorial(a){
    // if value of a is 1 then store the value as 1 and then return
    if(a == 1){
       fval = 1
    }else{
        // come here and call  the fnction again with value a - 1
        z = a - 1
            fnew = factorial(z)
            fval = a * fnew
    }
    // return the value from here
    return fval

}


method main(void){

    a = 6
    print("Computing the factorial of value : ")
    print(a)
    fact = factorial(a)
      print("Factorial value is : ")
      print(fact)

}
```

2. **Intermediate code generated:**

```
FuncDef factorial
STORE a

.funcBodyStarts factorial
PUSH 1
LOAD a
EQ

IFtrue label#1
PUSH 1
STORE fval
Go-Endifelse label#1
IFfalse label#2
PUSH 1
LOAD a
SUB
```

```
STORE z

FuncCall factorial
LOAD z
FuncCall Ends

STORE fnew
LOAD fnew
LOAD a
MUL
STORE fval
EndIfelse label#2
LOAD fval
.funcbodyends

.MainMethodStarts
PUSH 6
STORE a
PRINT "Computing the factorial of value : "
PRINT a

FuncCall factorial
LOAD a
FuncCall Ends

STORE fact
PRINT "Factorial value is : "
PRINT fact

.MainMethodEnds
```

3. <u>Generated Output:</u>

Steps to Execute:

a) First we have to compile the program file (*.bru). Copy the **compile.jar** and **runtime.jar** in a directory and create a filename with a .bru extension in the same directory. The command to compile the program is :

bruc filename.bru -> This generates a file with a ".bruclass" extension.

In this case the file generated will be ***filename.bruclass***

b) The second step is to execute the ".bruclass" generated in the above step. The command to execute the program is:

bru filename.bruclass

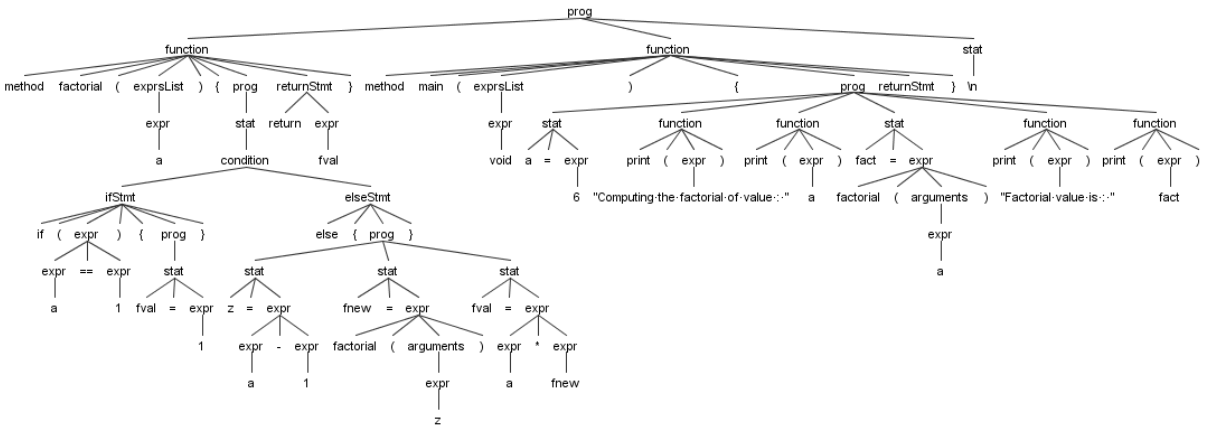
A snap of running the above two steps is attached below. First we compiled the file *"factorial.bru"* using the command in step a. This step created the intermediate file *"factorial.bruclass".* We then executed the class file by using the command in step b.

<u>Note:</u> Keep all the executable i.e. bru, bruc, jars, and the program files in the same directory

```
C:\Users\Abhinav\Desktop\Lang>bruc factorial.bru

C:\Users\Abhinav\Desktop\Lang>java -jar Compile.jar factorial.bru

C:\Users\Abhinav\Desktop\Lang>bru factorial.bruclass

C:\Users\Abhinav\Desktop\Lang>java -jar Runtime.jar factorial.bruclass
Computing the factorial of value :
6
Factorial value is :
720

C:\Users\Abhinav\Desktop\Lang>
```

---

4. <u>Parse Tree generation of the above program:</u>



---

5. <u>Some key points about our language:</u>

- <u>A variable is defined as:</u>

  a = 1

  b = false

- <u>A function is defined as:</u>

  method funcname(argList separated by comma) {

  // The the body goes over here

  }

- <u>Our main as it takes no argument is defined as:</u>

  method main(void){ //body over here}

- Functional can be called as:

  If it has a return value - > a = funname(var)
  If there is no return value -> funcname(var)

- If- else can be written as:
  If(condition){
  //body of if
  }else{
  //body of else
  }
- While can written as:
  while(condition){
  //body of while
  }
- Stack can be declared as:
  stack s -> declares a stack
  s.popstk() -> pops the value from stack and won't return
  s.peekstk() - > returns the top value from stack
  s.emptystk() -> return true or false on whether the stack is empty or not respectively
  s.pushstk(a) -> pushes the value of onto stack.

NOTE:

To execute the sample programs on windows platform given along with the zip:

- Open command prompt and go to the directory called ***src*** which contains the files i.e. compile.jar, runtime.jar, program files with **.bru** extension and two executables i.e. **bru** and **bruc.**
- Compile the program (*.bru) that you want to execute by typing the command on cmd:

  bruc factorial.bru
- Execute the intermediate code by typing the command on cmd as follows:

  bru factorial.bruclass
- The output will be displayed on the cmd.