**CS 341: Foundations of Computer Science II**
**Arashdeep Kaur**

# Project 2

Be sure to read this entire document before starting the assignment.

## Academic Integrity

Any student caught cheating on any assignment will be reported to the dean of students. Cheating includes, but is not limited to, getting solutions (including code) from, or giving solutions to someone else. You may discuss the assignment with others, but ultimately you must do and turn in your own work.

## Late submission Policy:

Homework assignments or projects or any other course related work assigned are expected to be submitted by the specified due date and time. Late submissions will only be considered within a specific window of time, which is up to 5 days after the original due date. For each day an assignment is submitted late, a penalty of 10% of the total possible grade for that assignment will be deducted. This deduction will accumulate for each additional day of late submission. The maximum penalty for late submissions is 50% of the total possible grade for the assignment. After the 5-day late submission window has passed, no further submissions will be accepted. It is the responsibility of the student to manage their time effectively and submit assignments within the specified timeframe.

## 1 Overview

The language $A$ is a particular set of strings (defined below) that represent valid arithmetic expressions operating on floating-point numbers, with the entire expression contained between two strings of symbols of a specified form. For this assignment you are to submit the formal definition (7-tuples) and a state diagram of a PDA that recognizes this language, a program that implements your PDA, and your code's output (following a specified format) on the provided test cases.

## 2 The Language $A$

The language $A = L(G)$ is defined as the set of strings where each string in $A$ is of the form $\mathbf{ab^ka}\boldsymbol{E}\mathbf{ab^ka}$ for any $k \geq 0$, where $E$ is an arithmetic expression over floating-point numbers, as defined by the set of production rules of the corresponding Context free grammar. Consider the following context free grammar G for the language A:

$$G = (V_N, V_T, P, S),$$

where,

$\boldsymbol{V_N} = \{S, T, C, H, Y, N\}$ is the set of non-terminals (variables);

$\boldsymbol{V_T} = \{ ., 0, 1, 2, \ldots, 9, +, -, *, /, (, ), a, b \}$, the set of terminal symbols *(1)*

which includes a dot for float-point numbers;

the starting variable is $\boldsymbol{S}$; and

**P**: the set of production rules are

$$\begin{aligned}
S &\to \mathrm{a}T\mathrm{a} \\
T &\to \mathrm{b}T\mathrm{b} \mid \mathrm{a}C\mathrm{a} \\
C &\to C{+}C \mid C{-}C \mid C{*}C \mid C/C \mid (C) \mid H \\
H &\to Y.Y \mid Y. \mid .Y \\
Y &\to NY \mid N \\
N &\to 0 \mid 1 \mid 2 \mid \cdots \mid 9
\end{aligned}$$

Let us consider the string w= "abbba(15.-(6.312*.7))abbba" and check whether this string can be generated by the above CFG or not.

This, we can show by using the derivation with $G$ *as follows*:

$S \Rightarrow$ a$T$a $\Rightarrow$ ab$T$ba $\Rightarrow$ abb$T$bba $\Rightarrow$ abbb$T$bbba $\Rightarrow$ abbba$C$abba

$\Rightarrow$ abbba($C$)abbba $\Rightarrow$ abbba($C$-$C$)abbba

$\Rightarrow$ abbba($C$-($C$))abbba $\Rightarrow$ abbba($C$-($C$*$C$))abbba

$\Rightarrow$ abbba($H$-($C$*$C$))abbba $\Rightarrow$ abbba($Y$.-($C$*$C$))abbba

$\Rightarrow$ abbba($NY$.-($C$*$C$))abbba $\Rightarrow$ abbba($NN$.-($C$*$C$))abbba

$\Rightarrow$ abbba(1$N$.-($C$*$C$))abbba $\Rightarrow$ abbba(15.-($C$*$C$))abbba

$\Rightarrow$ abbba(15.-($H$*$C$))abbba $\Rightarrow$ abbba(15.-($Y$.$Y$*$C$))abbba

$\Rightarrow$ abbba(15.-($N$.$Y$*$C$))abbba $\Rightarrow$ abbba(15.-(6.$Y$*$C$))abbba

$\Rightarrow$ abbba(15.-(6.$NY$*$C$))abbba $\Rightarrow$ abbba(15.-(6.$NNY$*$C$))abbba

$\Rightarrow$ abbba(15.-(6.$NNN$*$C$))abbba $\Rightarrow$ abbba(15.-(6.3$NN$*$C$))abbba

$\Rightarrow$ abbba(15.-(6.31$N$*$C$))abbba $\Rightarrow$ abbba(15.-(6.312*$C$))abbba

$\Rightarrow$ abbba(15.-(6.312*$H$))abbba $\Rightarrow$ abbba(15.-(6.312*.$Y$))abbba

$\Rightarrow$ abbba(15.-(6.312*.$N$))abbba $\Rightarrow$ abbba(15.-(6.312*.7))abbba

## 3 PDA for $A$

First you are to construct a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ that recognizes $A$, where $\Sigma$ is defined in equation (1).

In this project, the machine you design needs to only *recognize* the language $A$, not to *evaluate* the arithmetic expression.

## 4 Program Specifications

You must write your program in C, C++, Java, or Python. All input/output must be through standard input/output, and your program is to work as follows:

1. Your program first prints:

   Project 2 for CS 341

   Section number: *the section number you are enrolled in*

   Semester: Spring 2024

   Written by: *your first and last name, your NJIT UCID*

   Instructor: Arashdeep Kaur, ak3257@njit.edu

   where your NJIT UCID is typically your initials followed by some digits.

2. Your program asks the user to enter an integer $n \geq 0$ specifying the number of input strings to be processed, and your program prints out the value of $n$. If $n = 0$, the program terminates. If the user specified $n \geq 1$, your program enters a loop indexed by $i = 1, 2, \ldots, n$.

3. In the $i$th iteration of the loop, your program prompts the user, "Enter string $i$ of $n$", where $i$ is the iteration number and $n$ is the total number of strings to enter, and your program then reads in the string. You may assume that the user will only enter a string over $\Sigma$. After reading in the string, your program prints it. Then your program processes the string on your PDA in the following manner.

   - Your program must begin in the start state of the PDA and print out the name of that state ($q_1$ or $q_0$).
   - After each transition of the PDA, your program must print out
     - the state of the PDA before taking the transition,
     - the symbol from $\Sigma$ that was read,
     - the symbol from $\Gamma_\varepsilon$ that was popped from the stack,
     - the symbol from $\Gamma_\varepsilon$ that was pushed onto the stack, and
     - the state of the PDA after completing the transition.

– Sample format for state transition:

  - ➢ Present State: *q*
  - ➢ Current input symbol under R-head: *a* (it can be epsilon also)
  - ➢ Stack Top: *S* (it can be epsilon, in case stack is empty)
  - ➢ Symbol popped from Stack: *a* (it can be epsilon also, in case no pop)
  - ➢ Symbols pushed onto Stack: *b* (it can be epsilon also, in case no push)
  - ➢ Next state: *q1*

- If the PDA crashes before reaching the end of the input string, your program should output this fact: *String w= "print the string" is not acceptable by the given PDA because "print the reason here for non-acceptance".* Recall that a string is accepted by PDA if it reaches the final state and the whole input has been parsed and the stack is empty or it can contain only the special symbol (in this case S).
- After completing the processing of the current string on the PDA, your program must indicate if the string is accepted or rejected based on how the string was processed on the PDA.

4. After processing the $n$th string, your program terminates.

5. For all functions, subroutines, and classes you define in your code, their names must end with the last 3 digits of your UCID.

## 5 Test Cases

Test your program on each of the following input strings:

1. abbba43.51386abbba
2. aa.78+27.-3.013/837.842+aa
3. aa48622.+.41*1.2/00.1/521.23-.9+.53/7.aa
4. abba382.89*14.2aba
5. aba4.91-.*17.9aba
6. aa44.88.6+3.208aa
7. aba(1.2+(3.5-.9)/19).3aba

8. abba(.4)64abba
9. aba((824.23+(9.22-00.0)*21.2))+((.2/7.))abba
10. aba(())aba
11. abba((14.252+(692.211*(.39+492.1))/49.235)abba
12. abba+6.5abba
13. aa26.0*(.87/((4.+.2)/(23.1)-2.9)+6.)/(((823.*.333-57.*8.0)/.33))+.76aa
14. abba.0*(32.922+.7-*9.))abba
15. aba(4.+(.8-9.))/2.)*3.4+(5.21/34.2aba

You must create an output file containing your program's output from each test case in the order above.

## 6 Deliverables

You must submit all the following through Canvas by the due date/time:

1. A Microsoft Word document stating, "I certify that I have not violated the University Policy on Academic Integrity", followed by your first and last name, NJIT student ID, and UCID. If you do not include this pledge, then you will receive a 0 on the assignment. Anyone caught violating the University Policy on Academic Integrity will be reported to the dean of students.
2. Formal definition of PDA with all the tuples defined properly equivalent to the given grammar G and a state diagram of the PDA for *G* that your program implements. This format of the file must be either Microsoft Word, pdf, or jpeg (e.g., a photo from your phone's camera, but make sure it's not blurry). The file must be smaller than 5MB in size.
3. A **single file** of your source code, of type .c, .cpp, .java, or .py. Only submit the source code; do not include any class files. You must name your file

   p2 24s 006 ucid.ext

   where 006 is your section number in which you are enrolled, ucid is replaced by your NJIT UCID (which is typically your initials followed by some digits), and .ext is the appropriate file extension for the programming language you used, e.g., .java. The first few lines of your source code must be comments that include your full name, UCID, section number, and semester/year.
4. A **single file** containing the output (in the format specified in Section 4) from running your program on all of the test cases, in the order given in Section 5 of this document. The output file must be either .txt or in Microsoft Word.
5. The files **must not be compressed**.

## 7 Grading Criteria

The criteria for grading are as follows:

- the correctness and completeness of the tuples for the given grammar G. (35 points)
- the correctness of transition diagram of your PDA for the given grammar G. (20 points)
- your program follows the specifications given in Section 4, matches your PDA for $A$, and follows the directions in Section 6 (20 points)
- your program is properly documented with comments explaining how each block of your code works (5 points)
- your output is correct for the test cases and in the proper format as specified in section 4 (20 points).

Your grade will mainly be determined by examining the source code, the PDA, and the output that you turn in; the source code will only be tested if there are questions about your code.

To receive any credit for this assignment, you must turn in a PDA that your program implements and a *minimally working* program. For a program to be minimally working, it must

- compile without syntax errors;
- properly process all strings in $A_0$, where $A_0$ is the set of strings in $A$ that can accept at least $\mathbf{ab}^k\mathbf{aab}^k\mathbf{a;}$ and
- implement the drawing of your PDA for $A$.

**If you do not hand in a minimally working program, then you will receive a 0 for the assignment.**