Sabancı University
Faculty of Engineering and Natural Sciences

CS305 – Programming Languages

MIDTERM I

March 25, 2021

PLEASE NOTE:

- There are 3 questions in this exam.
- Provide only the requested information and nothing more.
- Unreadable, unintelligible and irrelevant answers will not be considered.

Question 1) [5 points] Consider a stack on which a sequence of `PUSH` and `POP` operations is performed.

(1) The stack is initially empty.

(2) The stack is also empty at the end.

(3) The stack may become empty during the operations as well, but when it is empty `POP` is not executed. In other words, we never `POP` from an empty stack.

Some example possible sequences of `PUSH` and `POP` operations are given below (each line is a separate example):

```
PUSH POP
PUSH POP  PUSH POP
PUSH POP  PUSH POP  PUSH POP
PUSH PUSH POP  POP
PUSH PUSH PUSH POP  POP  POP
PUSH PUSH POP  PUSH POP  POP  PUSH PUSH POP  POP
```

Consider a context free grammar G such that any complete sequence of `PUSH` and `POP` operations that can be performed on this stack is in the language of G. So, all the lines in the examples given above will be in the language of this grammar, along with any other possible sequence of `PUSH` and `POP` operations that can be performed on this stack.

a) (3pts) Design an ambiguous context free grammar G using:
   - only one nonterminal `<S>`,
   - only two tokens `tPUSH` (for `PUSH` operations) and `tPOP` (for `POP` operations)
   - at most 3 productions

b) (2pts) Show that your grammar is ambiguous by using a sentence with at most 6 tokens.

Question 2) [5 points] Write a complete flex file that will remove the comments in a C source file. The C source will be read by your flex translator and the same C file will be printed out with all the comments removed (anything outside a comment must be passed to the output as is).

Consider line comments (i.e. `//...` style comments) only. Assume that there are no block comments (i.e. `/* ... */` style comments) used in the file.

Basically, whenever the comment start sequence `//` is seen on a line, both `//` and anything that follows it on the same line upto the newline character should be removed. However, the newline character at the end of the line should be preserved.

Example executions

Before	After

`int x = 0; // decl of x`
`int y = 5; // this is another comment`
	`int x = 0;`
	`int y = 5;`

`int x = 0; // decl of x`
`// x = 5;  // don't need this for the time being`
`// also another declaration takes place here`
`int y = 5; // decl of y`
	`int x = 0;`
	
	`int y = 5;`

Question 3) [**5 points**] Consider a non-empty list of integers which is surrounded by a pair of brackets, such as `[3 5 -12 4]`, or `[23]`, or `[-33 -5 -5 14]`. Let us call such a list as "ordered" if the numbers in it are given in a nondecreasing order. So `[23]` and `[-33 -5 -5 14]` are ordered but `[3 5 -12 4]` is not ordered.

The following grammar which can be used to parse such a list of integers:

```
<intList> -> tLBRACKET <numbers> tRBRACKET
<numbers> -> tNUMBER <numbers>
<numbers> -> tNUMBER
```

We assume that we have a scanner that will give us the tokens `tLBRACKET` and `tRBRACKET` whenever it sees a left bracket and a right bracket, respectively. The scanner also returns the token `tNUMBER` when it sees an integer number in the input.

Suppose that we would like to check whether an integer list parsed by `<intList>` is ordered or not. For this purpose, we decide to use an attribute grammar. We consider the use of the following attributes associated with the grammar symbols:

- A boolean attribute named `ordered` for `<intList>`: It will tell whether the input parsed by `<intList>` is ordered or not. This attribute will be assigned by a semantic action.
- A boolean attribute named `ordered` for `<numbers>`: It will tell whether the input parsed by `<numbers>` is ordered or not. This attribute will be assigned by a semantic action.
- An integer attribute named `firstNumber` for `<numbers>`: It will keep the value of the first integer in the sequence parsed by this `<numbers>`. This attribute will be assigned by a semantic action.
- An integer attribute named `value` for `tNUMBER`: It will keep the value of the integer corresponding to this token. This attribute is assigned by the scanner.

Write the semantic actions associated with the productions to accomplish this task:

`<intList> -> tLBRACKET <numbers> tRBRACKET`	`.....<--- SEMANTIC ACTION 1`
`<numbers> -> tNUMBER <numbers>`	`.....<--- SEMANTIC ACTION 2`
`<numbers> -> tNUMBER`	`.....<--- SEMANTIC ACTION 3`

(Hint: As you are trying to answer this question, to see the semantic actions required, considering a parse tree for a list with 2-3 numbers might help.)