

John Vincent S. Racimo
BSCS 3 - 2
2021-01335
DBTK

1. Simulate the expressions using your code (hand written).

Program 1:

$$a = 2, c = 1, e = 5, g = 4, h = 0$$

$$z = a * 4 \% c - 2 / e + 2 * g - h / 1$$

play(z)

Simulation:

$$z = (2 * 4) \% 1 - 2 / 5 + (2 * 4) - 0 / 1$$

$$z = 8 \% 1 - 2 / 5 + 8 - 0 / 1$$

$$z = 0 - 2 / 5 + 8 - 0 / 1$$

$$z = 0 - 0.4 + 8 - 0 / 1$$

$$z = 0 - 0.4 + 8 - 0$$

$$z = -0.4 + 8 - 0$$

$$z = 7.6 - 0$$

$$\boxed{z = 7.6}$$

Program 2:

$$a = 6, c = 2, f = 0, i = 5$$

$$y = 8 / 2 * a \% 3 + e - f * 4 / 2 + i$$

play(y)

Simulation:

$$y = 8 / 2 * 6 \% 3 + 2 - 0 * 4 / 2 + 5$$

$$y = 4 * 6 \% 3 + 2 - 0 * 4 / 2 + 5$$

$$y = 24 \% 3 + 2 - 0 * 4 / 2 + 5$$

$$y = 0 + 2 - 0 * 4 / 2 + 5$$

$$y = 0 + 2 - 0 / 2 + 5$$

$$y = 0 + 2 - 0 + 5$$

$$y = 2 - 0 + 5$$

$$y = 2 + 5$$

$$\boxed{y = 7}$$

Program 3:

$$a = 9, d = 5, e = 6, h = 7, i = 2$$

$$x = a - (4 / 2) + d \% e * 3 + 2 - (h / i)$$

play(x)

Simulation:

$$x = 9 - (4 / 2) + 5 \% 6 * 3 + 2 - (7 / 2)$$

$$x = 9 - 2 + 5 \% 6 * 3 + 2 - 2$$

$$x = 9 - 2 + 5 * 3 + 2 - 2$$

$$x = 9 - 2 + 15 + 2 - 2$$

$$x = 7 + 15 + 2 - 2$$

$$x = 22 + 2 - 2$$

$$x = 24 - 2$$

$$\boxed{x = 22}$$

2. Run your compiler and display the output to validate your simulation.

Program 1:

A terminal window with a dark background. The title bar is dark blue with the word "Terminal" in white. The main area is dark blue with the number "7.6" in white.

Terminal

7.6

Program 2:

A terminal window with a dark background. The title bar is dark blue with the word "Terminal" in white. The main area is dark blue with the number "7.0" in white.

Terminal

7.0

Program 3:

A terminal window with a dark background. The title bar is dark blue with the word "Terminal" in white. The main area is dark blue with the number "22.0" in white.

Terminal

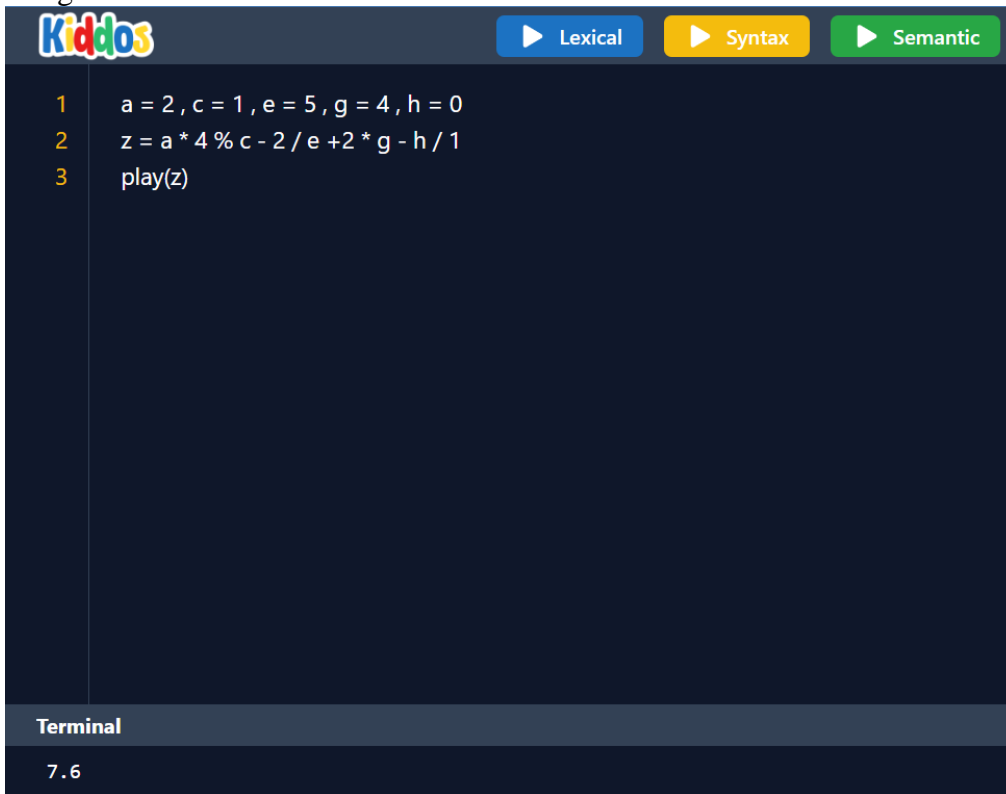
22.0

3. Submit the screenshot of your code.

```
400 def validateStringUsingStackBuffer(parsing_table, grammar_ll,
401                                   table_term_list, input_string,
402                                   term_userdef, start_symbol, token_lines, lexemes):
403
404     print("Validate String <=> (input_string)\n")
405
406     # for more than one entries
407     # - in one cell of parsing table
408     if grammar_ll == False:
409         return ("Input String = (input_string) Grammar is not LL(1)")
410
411     # implementing stack buffer
412     stack = [start_symbol, '$']
413     buffer = []
414
415     # reverse input string store in buffer
416     token_pos = 0
417     input_string_list = [(token[0], index) for index, token in enumerate(token_lines)]
418     input_string_list.reverse()
419     buffer = [('$', None)] + input_string_list
420
421     # Initialize an empty dictionary to store token frequencies
422     token_pos = 0
423     processed_tokens = []
424     py_processed_tokens = []
425     py_lexemes = [x for x in lexemes if x != '\n']
426
427     sample = []
428
429     # global_def_pos = 0
430     # is_global = True
431     # local_def_pos = 0
432     # global_list_pos = 0
433     # global_func_pos = 0
434     # current_state = None
435
436     # Initialize a flag variable outside the loop
437     first_definition_or_statement = True
438
439     brace_stack = []
440     is_call = False
441     pos_paramtail = 0
442     pos_call = 0
443     a = 0
444
445     while True:
446         # keep track of processed tokens
447         if buffer[-1][0] != '$' and (not token_pos or token_pos[-1] != buffer[-1][1]):
448             token_pos.append(buffer[-1][1])
449             processed_tokens.append(buffer[-1][0])
450             py_processed_tokens.append(buffer[-1][0])
451
452         # sample.append(buffer[-1][0])
453         # and keep if all symbols matched
454         if stack == ['$'] and buffer == ['$']:
455             # Replace the placeholders with newline characters
456             return ("Valid Syntax. (0) \n(processed_tokens) \n (lexemes) \n \n (py_processed_tokens) \n (py_lexemes) \n \n (sample)", py_lexemes)
457         elif stack[0] not in term_userdef:
458             # take front of buffer (y) and top (x)
459             if stack[0] in dictio:
460                 x = list(dictio.keys()).index(stack[0])
461             else:
462                 expected_tokens = firsts["PROGRAM"]
463                 if 'd' in expected_tokens:
464                     expected_tokens.remove('d')
465                 if 's' in expected_tokens:
466                     expected_tokens.remove('s')
467                 if 'd' in expected_tokens:
468                     expected_tokens.remove('d')
469                 return ("SyntaxError at Line (token_lines[len(processed_tokens)-1][1]) Invalid Token '{buffer[-1][0]}' Expected: {expected_tokens}")
470         y = table_term_list.index(buffer[-1][0])
471         if parsing_table[x][y] != '':
472             # format table entry received
473             entry = parsing_table[x][y]
474             lhs_rhs = entry.split('<->')
475             lhs_rhs[1] = lhs_rhs[1].replace('d', '').strip()
476             entryrhs = lhs_rhs[1].split()
477             stack = entryrhs + stack[1:]
478             sample.append(lhs_rhs[0])
479             sample.append(py_processed_tokens[-1])
480             # sample.append(buffer[-1][0])
481             # sample.append(token_tracker)
482
483             # Check for var_def_tail and const_def_tail
484             if lhs_rhs[0] in ["VAR_DEF_TAIL", "CONST_DEF_TAIL"] and buffer[-1][0] == ' ':
485                 if py_processed_tokens[-1] == ' ':
486                     # modify token
487                     py_processed_tokens[-1] = '\n'
488                     # modify lexemes
489                     py_lexemes[len(py_processed_tokens) - 1] = '\n'
490
491             # Check if a code block is formed
492             if lhs_rhs[0] in ["FUNC_DEF", "FUNC_FOR_LOOP", "FOR_LOOP", "WHILE_LOOP", "FUNC_WHILE_LOOP", "CONDIF_STMT", "FUNC_CONDIF_STMT", "LOOP_CONDIF_STMT", "FUNC_LOOP_CONDIF_STMT"]:
493                 brace_stack.append('(')
494
495             # /> conversion
496             if lhs_rhs[0] in ["PARAMETER", "PARAM_TAIL", "RANGE_END", "RANGE_STEP", "INT_EXPR_TAIL", "EXPR_TAIL"] and buffer[-1][0] == ' ':
497                 pos_paramtail = len(py_processed_tokens)
498
499             if pos_paramtail != 0 and lhs_rhs[0] in ["FUNC_BODY", "LOOP_BODY", "FUNC_LOOP_BODY", "CONDIF_BODY"]:
500                 py_processed_tokens[pos_paramtail] = ' '
501                 py_lexemes[pos_paramtail] = ' '
502                 pos_paramtail = 0
503
504             # Add indentation to code block contents
505             if lhs_rhs[0] in ["FUNC_BODY", "FUNC_MORE_BODY", "LOOP_BODY", "LOOP_MORE_BODY", "FUNC_LOOP_BODY", "FUNC_LOOP_MORE_BODY", "CONDIF_BODY", "CONDIF_MORE_BODY"]:
506                 index = len(py_processed_tokens) - 1
507                 indents = '\n' * (len(brace_stack) if len(brace_stack) != 0 else 1)
508                 py_processed_tokens.insert(index, indents)
509                 py_lexemes.insert(index, indents)
510
511             # Remove closing brace
512             if buffer[-1][0] == ')':
513                 py_processed_tokens[-1] = ' '
514                 py_lexemes[len(py_processed_tokens)-1] = ' '
515                 if brace_stack:
516                     brace_stack.pop()
517                     a+=1
518
519             # Remove call . from FUNC_CALL
520             if lhs_rhs[0] == "FUNC_CALL":
521                 py_processed_tokens.pop()
522                 py_lexemes.pop(len(py_processed_tokens))
523                 py_lexemes.pop(len(py_processed_tokens))
524                 pos_call = len(py_processed_tokens)
525                 is_call = True
526
527             if is_call and lhs_rhs[0] == "ARGUMENT":
528                 py_processed_tokens.pop(pos_call)
529                 pos_call = 0
530
531             # Add /> to def and statements outside code blocks
532             if len(brace_stack) == 0 and lhs_rhs[0] in ["DEFINITIONS", "STATEMENTS"]:
533                 if not first_definition_or_statement:
534                     py_processed_tokens.insert(-1, '\n')
535                     index = len(py_processed_tokens) - 2
536                     py_lexemes.insert(index, '\n')
537                 else:
538                     first_definition_or_statement = False
539
540     first_definition_or_statement = False
```

4. Submit the screenshot of your output after running your compiler.

Program 1:



The screenshot shows the Kiddos IDE interface. At the top left is the 'Kiddos' logo. To its right are three buttons: 'Lexical' (blue), 'Syntax' (yellow), and 'Semantic' (green). Below these is a dark blue editor area containing three lines of code, each preceded by a line number (1, 2, 3) in yellow. The code is: `a = 2 , c = 1 , e = 5 , g = 4 , h = 0`, `z = a * 4 % c - 2 / e + 2 * g - h / 1`, and `play(z)`. At the bottom is a 'Terminal' panel with the text '7.6'.

```
1  a = 2 , c = 1 , e = 5 , g = 4 , h = 0
2  z = a * 4 % c - 2 / e + 2 * g - h / 1
3  play(z)
```

Terminal

7.6

Program 2:



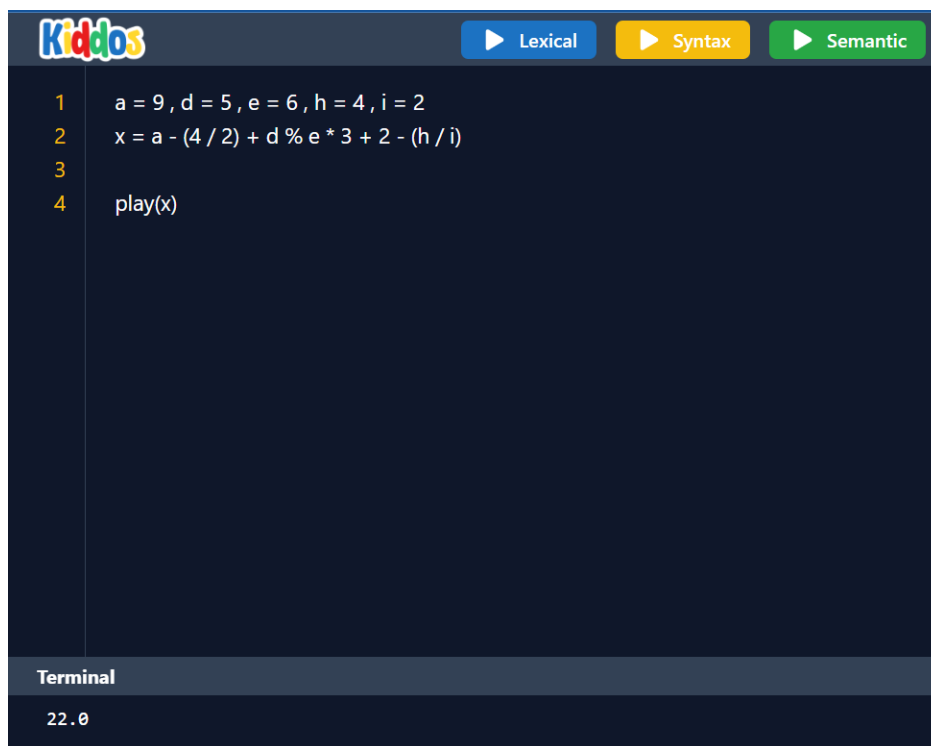
The screenshot shows the Kiddos IDE interface. At the top, there is a header bar with the 'Kiddos' logo on the left and three buttons: 'Lexical' (blue), 'Syntax' (yellow), and 'Semantic' (green). The main editor area has a dark blue background and contains three lines of code, each preceded by a line number in yellow: 1. `a = 6 , e = 2 , f = 0 , i = 5`, 2. `y = 8 / 2 * a % 3 + e - f * 4 / 2 + i`, and 3. `play(y)`. At the bottom, there is a 'Terminal' panel with a dark blue background, displaying the text '7.0'.

```
1 a = 6 , e = 2 , f = 0 , i = 5
2 y = 8 / 2 * a % 3 + e - f * 4 / 2 + i
3 play(y)
```

Terminal

7.0

Program 3:



The screenshot shows the Kiddos IDE interface. At the top, there is a header bar with the 'Kiddos' logo on the left and three buttons: 'Lexical' (blue), 'Syntax' (yellow), and 'Semantic' (green). The main editor area has a dark blue background and contains four lines of code, each preceded by a line number in yellow: 1. `a = 9 , d = 5 , e = 6 , h = 4 , i = 2`, 2. `x = a - (4 / 2) + d % e * 3 + 2 - (h / i)`, 3. (empty line), and 4. `play(x)`. At the bottom, there is a 'Terminal' panel with a dark blue background, displaying the text '22.0'.

```
1 a = 9 , d = 5 , e = 6 , h = 4 , i = 2
2 x = a - (4 / 2) + d % e * 3 + 2 - (h / i)
3
4 play(x)
```

Terminal

22.0