| | |
|---|---|
| **Started on** | Thursday, 3 August 2023, 14:05 |
| **State** | Finished |
| **Completed on** | Thursday, 3 August 2023, 14:30 |
| **Time taken** | 25 mins |
| **Grade** | **11.8** out of 20.0 (**58.8%**) |

Question **1**
Partially correct
Mark 0.8 out of 3.0
Flag question

In Prac 3 you made use of the Coco runtime system to create parsers from grammar files, e.g. Calc.atg. To simplify this process you were given various batch files to execute

Fill in the missing words in the sentence below which explains how Coco is used to create a parser for the Calc.atg grammar.

After creating the Calc.atg grammar, the [asm.bat] ✗ script is used to generate a working parser. The first executable program called by this script is [coco.exe] ✗. After executing this program, various [Java source files] ✓ are created and placed in a folder called [Assem] ✗.

Your answer is partially correct.
You have correctly selected 1.
The correct answer is:
In Prac 3 you made use of the Coco runtime system to create parsers from grammar files, e.g. Calc.atg. To simplify this process you were given various batch files to execute
Fill in the missing words in the sentence below which explains how Coco is used to create a parser for the Calc.atg grammar.
After creating the Calc.atg grammar, the [cmake.bat] script is used to generate a working parser. The first executable program called by this script is [coco.jar]. After executing this program, various [Java source files] are created and placed in a folder called [Calc].

Question **2**
Complete
Mark 3.0 out of 4.0
Flag question

In prac 3 you created a Cocol token for a floating point number. Now create a Cocol token for a number that will accept the following forms:

123 1.2E12 1.02E-12 1E-6 1.01

You do not have to accept numbers with no digits before the "." .23 .2E1

```
CHARACTERS
   digit = "0123456789" .
TOKENS
   number =

CHARACTERS
   digit = "0123456789"
TOKENS
   number = digit {digit} [sci]
   sci = ["."] (digit) {"E"} ["-"] digit {digit}
```

Comment:
not quite as these tokens are not distinct. How do we know when to move to Sci as the starting part of this could be just a digit?

Question **3**
Correct
Mark 1.0 out of 1.0
Flag question

What is an ASCII table? Choose the most comprehensive answer.

Select one:

- It contains a list of printable and non-printable characters used in programming
- It contains a list of all characters used in programming together with an integer representation of each character, sorted so that the printable characters have low integer representations, and the non-printable characters have higher integer values.
- It contains a list of all characters, both printable and non-printable, used in programming together with a hexadecimal representation of each character. ✓
- It contains a list of all printable characters used in programming together with a hexadecimal representation of each character

Your answer is correct.
The correct answer is:
It contains a list of all characters, both printable and non-printable, used in programming together with a hexadecimal representation of each character.

Question **4**
Complete
Mark 3.0 out of 4.0
Flag question

In pracs 2 and 3 you added several new opcodes to the PVM interpreter. You are now asked to add an opcode (PVM.max) which calculates the maximum of two operands. This opcode makes use of the top two operands on the stack, calculates the maximum value of these and leaves this result on the top of the stack (in the same way as other binary operators do).

Write the code (you can use Java or pseudocode) that allows the PVM interpreter to "execute" this opcode. Your code will be inserted into the main switch statement of the PVM interpreter, using a case statement like:

```
case PVM.max:   // new code goes here
                break;


case PVM.max:
    tos = pop() ///Assumes the value is stored on stack and not the address
    sos = pop()
    if (tos == sos){
        push(tos) //They both equal so it doesn't matter
        break;
    else if(tos > sos){
        push(tos) //tos > sos & tos != sos as that check was done first
        break;
    else if(sos > tos){
        push(sos); //Only other case is sos > tos besides errors
        break;
    else{
        ps = badMem;
        break;
    break;
```

Comment:
There cannot be any alternatives other than >, = or <. Not sure why this is a bad memory issue. Keep it simple is the motto in this class, but your code has been over-thought hugely.

Question **5**
Complete
Mark 1.0 out of 2.0
Flag question

In the grammar you created to parse the index list, it is important to identify the end-of-line character. Why?

If we do not recognise the end of line character (represented by string "\n") then our grammar will come across the hidden \n and break. Just because we don't see it doesn't mean it doesn't exist. In our other programs this was not an issue as there was an identifier for when the end of line is such as in songs we had the "." but for index we have nothing besides a new line which is represented by our end of line character.

Comment:
Not true – a newline char is treated the same way as spaces -- i.e. the parser skips over them if you don't explicitly ask the parser to consider them.

Question **6**
Incorrect
Mark 0.0 out of 2.0
Flag question

When defining Cocol grammars, we need to pay attention to spaces, the end of line character, and the end of file character.

Which of the following statements is/are true:

1. A scanner generated using Coco will always ignore all spaces and end of line characters when creating tokens.
2. A parser generated using Coco will ignore all spaces when applying productions.
3. The end of line character can be represented by the token EOL which is defined as CHR(10).
4. EOF represents the end of file character and is used in a production to signal that the input has been successfully parsed and to tell the parser that it can stop parsing.

Select one:

- Statements 1 and 3 are true
- All of the statements are true ✗
- Statements 2 and 3 are true
- None of the statements is true
- Statements 2, 3 and 4 are true

Your answer is incorrect.
The correct answer is:
Statements 2 and 3 are true

Question **7**
Complete
Mark 3.0 out of 4.0
Flag question

In prac 3 you created a grammar to parse a list of songs. Given below is a possible solution.

EXPLAIN in your own words whether this grammar would successfully parse the song list (some examples are given below). If it can parse these, explain how it does this, and if it does not parse the songs, explain why not.

Example songs:

"Shivers" (Ed Sheeran) [2022 ] .
"The crossing" (Friends of Johnny Clegg) .

```
// Newsong.atg
COMPILER Newsong $CN

CHARACTERS
   letter = 'A'..'Z' + 'a'..'z'+ "" + "'-[](). ".
   digit = "0123456789".

TOKENS
   word = letter {letter|digit} .

IGNORE CHR(0) .. CHR(31)

PRODUCTIONS
   Newsong = {word} EOF.

END Newsong.
```

This production would parse correctly.
If we had to draw a parse tree for this grammar it would go something like
Newsong -> word -> letter {letter|digit} -> the characters for both letter and digit.
Each one of the characters in both lines are in the character sets for letter and digit (including a "."). And then we are ignoring CHR(0) .... CHR(31), meaning we ignore any new line characters and thus our example songs would be read as if they are all one line. This means that our songs would parse but they wouldn't be structured, we would just have a long list of words.

Comment:

Finish review

Quiz navigation

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|

Show one page at a time
Finish review