EXAM PAPERS PRACTICE

3.3 Reverse Polish

Name: ____________________

Class: ____________________

Date: ____________________

Time: **36 minutes**

Marks: **27 marks**

Comments:

Q1.

How would the infix expression `5 – 3` be represented in Reverse Polish notation?

____

____

(Total 1 mark)

Q2.

How would the infix expression `3 + 4 * 2 – 1` be represented in Reverse Polish notation?

____

____

____

____

(Total 2 marks)

Q3.

Explain why Reverse Polish notation is sometimes used instead of infix notation.

____

____

____

____

(Total 2 marks)

Q4.

To evaluate an expression in Reverse Polish notation, you start from the left hand side of the expression and look at each item until you find an operator (eg + or −).

This operator is then applied to the two values immediately preceding it in the expression. The result obtained from this process replaces the operator and the two values used to calculate it. This process continues until there is only one value in the expression, which is the final result of the evaluation.

For example `5 2 7 + +` would change to `5 9 +` after the first replacement.

Explain how a stack could be used in the process of evaluating an expression in Reverse Polish notation.

____

____

____

____

____

____

(Total 3 marks)

Q5.

Reverse Polish Notation is an alternative to standard infix notation for writing arithmetic expressions.

(a) Convert the Reverse Polish Notation expressions in the table to their equivalent infix expressions.

Reverse Polish Notation	Equivalent Infix Expression
18 9 –	
10 4 – 12 ×	

(2)

(b) State **one** advantage of Reverse Polish Notation over infix notation.

____

____

____

(1)

(Total 3 marks)

Q6.

Convert the following Reverse Polish Notation expressions to their equivalent infix expressions.

(a) `3 4 *`

____

____

(1)

(b) `12 8 + 4 *`

____

____

(1)

(c) Reverse Polish Notation is an alternative to standard infix notation for writing arithmetic expressions.

State **one** advantage of Reverse Polish Notation over infix notation.

____

___

(1)
(Total 3 marks)

Q7.

Reverse Polish Notation is an alternative to standard infix notation for writing arithmetic expressions.

(a) Convert the following Reverse Polish Notation expressions to their equivalent infix expressions.

Reverse Polish Notation	Equivalent Infix Expression
45 6 +	
12 19 + 8 *	

(2)

(b) State **one** advantage of Reverse Polish Notation over infix notation.

___

___

(1)

(c) The pseudo-code algorithm below can be used to calculate the result of evaluating a Reverse Polish Notation expression that is stored in a string. The algorithm is designed to work only with the single digit denary numbers 0 to 9. It uses procedures and functions listed in the table below, two of which operate on a stack data structure.

```
StringPos ← 0
Repeat
  StringPos ← StringPos + 1
  Token ← GetCharFromString(InputString, StringPos)
  If Token = '+' Or Token = '-' Or Token = '/' Or Token = '*'
    Then
      Op2 ←Pop()
      Op1 ← Pop()
      Case Token Of
        '+': Result ← Op1 + Op2
        '-': Result ← Op1 - Op2
        '/': Result ← Op1 / Op2
        '*': Result ←Op1 * Op2
      EndCase
      Push(Result)
    Else
      IntegerVal ←ConvertToInteger(Token)
      Push(IntegerVal)
  EndIf
Until StringPos = Length(InputString)
Output Result
```

Procedure/Function	Purpose	Example(s)
GetCharFromString (InputString:String	Returns the character	GetCharFromString

, StringPos:Integer): Char	at position StringPos within the string InputString. Note that the leftmost letter is position 1, not position 0.	("Computing", 1) would return the character 'C'. GetCharFromString ("Computing", 3) would return the character 'm'.
ConvertToInteger (ACharacter: Char): Integer	Returns the integer equivalent of the character in ACharacter.	ConvertToInteger('4') would return the integer value 4.
Length (AString: String): Integer	Returns a count of the number of characters in the string AString.	Length("AQA") would return the integer value 3.
Push (ANumber: Integer)	Puts the number in ANumber onto the stack.	Push(6) would put the number 6 on top of the stack.
Pop (): Integer	Removes the number from the top of the stack and returns it.	X ← Pop() would remove the value from the top of the stack and put it in X.

(d) Complete the table below to trace the execution of the algorithm when InputString is the string: 64+32+*

In the Stack column, show the contents of the stack once for each iteration of the Repeat..Until loop, as it would be at the end of the iteration.

The first row and the leftmost column of the table have been completed for you.

StringPos	**Token**	**IntegerVal**	**Op1**	**Op2**	**Result**	**Stack**
0	-	-	-	-	-	
1						
2						

3						
4						
5						
6						
7						

(5)

Final output of algorithm: ____

(1)

(e) A programmer is going to implement the algorithm above in a programming language that does not provide built-in support for a stack data structure.

The programmer intends to simulate a stack by using a fixed length array of 20 integers named `StackArray` with indices running from `1` to `20` and an integer variable `TopOfStackPointer` which will be initialised to `0`.

Write a pseudo-code algorithm for the `Push` operation to push a value stored in the variable `ANumber` onto the stack.

Your algorithm should cope appropriately with any potential errors that might occur.

____

____

____

____

(4)
(Total 13 marks)