# Technical Job Interview Questions for Classic Rexx Programmers – with Answers

by Howard Fosdick 
www.RexxInfo.org 

You should be able to complete this test in about a half hour. This is a “from memory” test only – no cell phone or computer access allowed. The answers follow the test.

1. What do these two Rexx statements accomplish?

```
social_security_number = '499-55-1212'
PARSE VAR social_security_number comp1 '-' comp2 '-' comp3 .
```

2. What does TRACE A do? What does TRACE O do?

3. Name two of Rexx’s three *special variables*, and explain their purpose.

4. How many characters does the CHARIN function return by default? Can it move the read pointer without reading any characters?

5. Why do you code a SIGNAL instruction in a Rexx program?

6. What is the function of the ADDRESS instruction?

7. What is the function of the ADDRESS function?

8. What two Rexx instructions can be used to implement a Case construct?

9. How do you set the arithmetic precision – the number of significant digits for calculations – in a Rexx program?

10. What is the difference between the equal to ( = ) and strictly equal to ( == ) operators?

11. What is the *remainder divide* operator and why would you use it?

12. What would you encode to uninitialize (or “unassign”) two variables named **one** and **two**?

13. What is the difference between the PUSH and QUEUE instructions?

14. What instruction could you code to leave an endless loop (a “do forever” loop) and proceed to the next statement following the DO loop?

15. On a Windows server, you’ve been put in charge of project to migrate hundreds of classic Rexx programs to Open Object Rexx (ooRexx). All programs you will migrate strictly conform to the ANSI Rexx standard. Would you expect that:

    (A) Zero program changes will be necessary because ooRexx runs any classic Rexx program
    (B) The vast majority of programs will run without changes, but you could run into a few exceptions
    (C) You’ll have to rewrite most of the classic Rexx programs to run under ooRexx
    (D) ooRexx is not compatible with classic Rexx. You’ll have to recode all the programs.

16. What does the ARG function do when processing input(s) given by a Rexx command (that is, a Rexx script run from the command line) ?

    (A) Parses elements in the exact same as PARSE UPPER ARG
    (B) Returns the number of arguments passed from the command line into the program
    (C) Returns 1 if there were one or more arguments passed into the program from the command line, and 0 otherwise
    (D) Returns the entire command line as the user entered it

17. You're downloading a Rexx script from the web that you want to run on your computer. But you notice that your keyboard has no symbol for this one that occurs in the program: ¬ . What symbol could you code instead of ¬ ?

18. How do you code a binary string constant within a Rexx program?

19. Between the Rexx comparison, arithmetic, and concatenation operators, which has highest priority?

20. For the compound variable List.j, what is the stem, and what is the tail (also called the subscript or index)?

21. How can you override the default precedence order of operations in a Rexx statement?

22. What are the OR and EXCLUSIVE OR operators, and what is the difference in how they operate?

23. Why would you code the PROCEDURE EXPOSE instruction?

24. When is the NOVALUE condition trap raised?

25. When you code a Rexx statement in your program, how can you continue it to a second line?

26. Why would you use the DIGITS function?

27. When you issue a command to an external environment from your Rexx program, how do you know if that command worked?

28. You have an error number (error code), and you want to find out what it means. How can you do this from within a Rexx program?

29. Which of these is **not** one of the standard Rexx keywords for trapping errors: ERROR, FAILURE, HALT, NOTVALID, and SYNTAX?

30. In this program, which statement will display output that differs from that of the other two statements?

```
one = 'one' ;     two = 'two'
say (one)(two)   /* statement A  */
say one || two    /* statement B */
say one  two      /* statement C */
```

31. Does the PULL instruction alter input data? Does PARSE PULL?

32. What is the effect of the period in this statement?

```
PARSE VAR  my_string   item_one   item_two   .
```

33. How do you initialize all elements of the compound variable stem (aka, an array) named MY_ARRAY. to 0?

---

## --- ANSWERS ---

1. **Answer**: The first statement initializes the variable **social_security_number** to the character string value **499-555-1212**. The second statement parses it into its three component pieces (as separated by the internal dashes) and places the components into the three variables **comp1**, **comp2**, and **comp3**, respectively.

   In other words, **comp1** will contain **499**, **comp2** will contain **555**, and **comp3** will contain **1212**.

2. **Answer**: TRACE A traces all clauses before execution. TRACE O turns off tracing.

3. **Answer**: Three special variables are RC, RESULT, and SIGL.

RC contains a return code from a command, or a syntax error code.

RESULT contains a value passed back from a RETURN instruction in a subroutine. (If a RETURN is encoded without an operand, RESULT is set to unitialized.)

SIGL identifies the line number of last instruction that caused a jump to a label.

4. **Answer**: The CHARIN function reads one character by default.

   Yes, you can use CHARIN to position the read pointer without actually reading any characters. For example, CHARIN('TEXT_FILE',1,0) positions the read pointer to the start of TEXT_FILE without reading any characters.

5. **Answer**: To enable one of SIGNAL's supported error conditions, and optionally specify its associated routine. Or to disable that error condition.

   (It's not recommended, but some also use it to simulate the action of a GOTO statement.)

6. **Answer**: To set the execution environment for command(s) sent out from a Rexx program.

7. **Answer**: To tell your Rexx program what the current external command execution environment is.

8. **Answer**: IF and SELECT. (Note that WHEN is a keyword or clause within the SELECT statement; it is not a Rexx instruction.)

9. **Answer**: Use the NUMERIC instruction.

10. **Answer**: *Strict comparison* requires that two strings be identical (leading and trailing blanks count in the comparison.)

11. **Answer**: The operator is: // . You use it when you want to obtain the remainder of a division operation.

12. **Answer**: DROP one two
    Or you could encode a pair of DROP instructions: DROP one ; DROP two ;

13. **Answer**: PUSH adds an element to the queue or data stack in the order Last-In, First-Out (LIFO).
    QUEUE adds an element to the queue or data stack in the order First-In, First-Out (FIFO).

14. **Answer**: The correct answer is the LEAVE instruction. It alters the flow of control from within a DO loop to the statement following its END clause.

    Instructions like EXIT and RETURN will break out of the endless loop, but do not direct execution to the code following the DO loop. You could also use SIGNAL for this purpose but this is poor programming practice.

15. **Answer**: Generally speaking, ooRexx runs classic Rexx programs that conform to the ANSI-1996 Rexx standard without any changes. However, there are rare exceptions (ooRexx lacks a couple new items from that standard.) With "hundreds" of programs involved, you might well run into a few of the execptions. The best answer is B.

16. **Answer**: Answer (C) is correct. (A) is incorrect because it tells what the ARG **instruction** does, not the ARG **function**. (B) is incorrect because it tells what ARG does when encoded to parse arguments in a Rexx subroutine or function. It does not tell what happens when ARG processes command line arguments (parameters encoded when Rexx is invoked from the command line). (D) is just plain wrong.

17. **Answer**: ¬ is the "not" or negation symbol. If your system doesn't support it, just use a backslash instead: \ .
    For example, instead of coding "not equal" as ¬= you would code it as \= .

18. **Answer**: Code it as a string of 0's and 1's within single or double quote marks followed by the letter 'b' or 'B'.
    Examples: '0101'b or "0101"B .

19. **Answer**: Arithmetic.

20. **Answer**: the stem is **List.** (it includes the period), and the tail (or subscript or index) is **j**

21. **Answer**: Use parentheses. Enclose the higher precedence operations you want executed first within the parentheses.

22. **Answer**: OR is: | EXCLUSIVE OR is: &&
OR means "true if either term is true," while EXCLUSIVE OR means "true if either but not both terms are true."

23. **Answer**: You code PROCEDURE EXPOSE to ensure that only the specified list of variables is available (or "exposed") to a subroutine. The subroutine can both read and update any variable(s) that are exposed to it.

24. **Answer**: When the NOVALUE error condition is enabled, and a variable is referenced that has not yet been initialized.

25. **Answer**: Encode a comma ( , ) at the end of the line to be continued.

26. **Answer**: It returns the setting of NUMERIC DIGITS (it returns the numeric precision).

27. **Answer**: Inspect the value of the special variable RC. It contains the command's return code.

28. **Answer**: One way is to use the ERRORTEXT function. It returns the error text associated with a given error number.

29. **Answer**: NOTVALID. For ANSI-standard classic Rexx the trap conditions are ERROR, FAILURE, HALT, NOTREADY, NOVALUE, SYNTAX, and LOSTDIGITS.

30. **Answer**: The outputs from the program will be:

    onetwo
    onetwo
    one two

    The answer is statement (C). It outputs the two variables with an intervening space (or blank).

    Statement (A) is concatenation by abuttal, while statement (B) uses the concatenation operator. Both output the two variables with no intervening space (or blank).

31. **Answer**: PULL automatically translates characters to their uppercase equivalents. PARSE PULL does not alter the data by performing uppercase translation.

32. **Answer**: The variable **item_one** contains the first data element parsed from the string called **my_string**, **item_two** contains the second, and the period ignores all subsequent data items. If the period were not encoded, **item_two** would contain the second and all subsequent input data elements concatenated together.

33. **Answer**: MY_ARRAY. = 0

---

**Grading –**

25 or more questions answered correctly (about 75%) indicate a credible candidate. Less than 20 questions correct indicate a candidate who is not credible.

Thanks to Rexx Language Association members for their feedback on these questions including Frank, Shmuel, David, Wayne, Rob, Mark, Chip, & others.