2.7 Programming Practice: Writing and Refining Programs: Test Pack

Total Marks

OCR J277 · Calculators not allowed · about 25 minutes

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 A program should input a whole number and output whether it is odd or even.

Write an algorithm that inputs an integer and outputs "Even" if the number is exactly divisible by 2, or "Odd" otherwise.

(Total for Question 1 is 3 marks)

2 A string variable is set as: name = "Aditi"

(a) A string variable is set as: **(1)**
name = "Aditi"

State the value of LEN(name).

(b) Write one line of code that outputs the first three characters of name, using SUBSTRING(startIndex, numberOfChars, string). **(2)**

(Total for Question 2 is 3 marks)

© Revision Library | revisionlibrary.co.uk

3 Write a function called isValidPassword that takes a string parameter password and returns a Boolean: True if the password is at least 8 characters long AND contains at least one digit (0-9), or False otherwise.

(Total for Question 3 is 6 marks)

© Revision Library | revisionlibrary.co.uk

4 A student has written the following function, intended to search for a value target in an array numbers and return its index, or -1 if not found. When tested, the function sometimes returns the wrong result:

```
FUNCTION search(numbers, target)
    FOR i = 0 TO LEN(numbers) - 1
        IF numbers[i] == target THEN
            found = i
        ELSE
            found = -1
        ENDIF
    NEXT i
    RETURN found
ENDFUNCTION
```

(a) A student has written the following function, intended to search for a value target in an array numbers and return its index, or -1 if not found. When tested, the function sometimes returns the wrong result: **(2)**

```
FUNCTION search(numbers, target)
    FOR i = 0 TO LEN(numbers) - 1
        IF numbers[i] == target THEN
            found = i
        ELSE
            found = -1
        ENDIF
    NEXT i
    RETURN found
ENDFUNCTION
```

Identify why this function does not always work correctly, using an example to support your answer.

(b) Rewrite the function so that it works correctly for every case. **(3)**

(Total for Question 4 is 5 marks)

© Revision Library | revisionlibrary.co.uk

5 A student has written the following program, which is intended to calculate and output the average of a set of numbers entered by the user. The user enters -1 to stop entering numbers. However, the program does not work correctly if a negative number (other than -1) is entered by mistake, and it does not handle the case where no valid numbers are entered at all:

```
total = 0
count = 0
INPUT number
WHILE number != -1
    total = total + number
    count = count + 1
    INPUT number
ENDWHILE
average = total / count
OUTPUT average
```

(a) A student has written the following program, which is intended to calculate and output the average of a set of numbers entered by the user. The user enters -1 to stop entering numbers. However, the program does not work correctly if a negative number (other than -1) is entered by mistake, and it does not handle the case where no valid numbers are entered at all: **(2)**

```
total = 0
count = 0
INPUT number
WHILE number != -1
    total = total + number
    count = count + 1
    INPUT number
ENDWHILE
average = total / count
OUTPUT average
```

Explain one weakness of this program, other than the lack of comments, referring to what happens if the user enters -1 as their very first input.

© Revision Library | revisionlibrary.co.uk

(b) Refine the program so that: any negative number entered (not just -1) ends the input phase in the same way as -1, and the program outputs a suitable message instead of calculating an average if no valid numbers were entered. **(4)**

(Total for Question 5 is 6 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.7 Programming Practice: Writing and Refining Programs: Test Pack

Question 1

INPUT statement storing the number in a suitably named variable (oe) **B1**

correct selection using MOD 2 == 0 (or an equivalent DIV-based test) as the condition **B1**

correct OUTPUT of "Even" and "Odd" in the correct branches (oe; ignore minor syntax, accept any equivalent selection structure) **B1**

Answer:

```
INPUT number
IF number MOD 2 == 0 THEN
    OUTPUT "Even"
ELSE
    OUTPUT "Odd"
ENDIF
```

Question 2

(a) 5 **B1**

(a) Answer: 5

(b) correct OUTPUT statement calling SUBSTRING with name as the string argument **B1**

(b) correct start index and length arguments that return the first three characters (oe; accept either a 0-indexed or 1-indexed convention provided the result is the first three characters) **B1**

(b) Answer: OUTPUT SUBSTRING(0, 3, name)

Question 3

correct FUNCTION header with one string parameter, e.g. FUNCTION isValidPassword(password) **B1**

correct check of the length condition, LEN(password) ≥ 8 (oe) **B1**

a Boolean flag (e.g. containsDigit = False) initialised before searching for a digit **B1**

correct iteration through every character of the password (e.g. FOR i = 0 TO LEN(password) - 1, oe) **B1**

correct selection testing whether the current character is a digit (e.g. comparing the character against "0" to "9") and setting the flag to True when found **B1**

correct final RETURN combining both conditions with AND (RETURN length condition AND containsDigit, oe) **B1**

Answer:

```
FUNCTION isValidPassword(password)
    containsDigit = False
    FOR i = 0 TO LEN(password) - 1
        character = SUBSTRING(i, 1, password)
        IF character >= "0" AND character <= "9" THEN
            containsDigit = True
        ENDIF
    NEXT i
    RETURN LEN(password) >= 8 AND containsDigit
ENDFUNCTION
```

Question 4

(a) identifies that found is reassigned on every iteration, so its final value only reflects the last comparison made, not whether a match was found earlier in the array **B1**

(a) supporting example showing a case where the target is found early but a later non-matching element overwrites found back to -1 (e.g. numbers = [4, 9, 2], target = 4: found is set to 0 at i = 0, but then overwritten to -1 at i = 1 and i = 2) **B1**

(a) Answer: The variable found is reassigned inside every loop iteration, including the ELSE branch, so a correct match found early is overwritten by later non-matches. Example: numbers = [4, 9, 2], target = 4: at i = 0, found = 0 (a correct match), but at i = 1, numbers[1] = 9 does not equal 4, so found is reset to -1, and again at i = 2. The function returns -1 even though 4 is in the array at index 0.

(b) correct RETURN of i placed immediately inside the IF branch when a match is found, exiting the function immediately rather than storing it in a variable that could later be overwritten **B1**

(b) removal of the incorrect ELSE branch that reset found on every non-matching element **B1**

(b) correct RETURN of -1 placed after the loop, only reached if no match was found during any iteration **B1**

(b) Answer:

```
FUNCTION search(numbers, target)
    FOR i = 0 TO LEN(numbers) - 1
        IF numbers[i] == target THEN
            RETURN i
        ENDIF
    NEXT i
    RETURN -1
ENDFUNCTION
```


© Revision Library | revisionlibrary.co.uk

Question 5

(a) identifies that if -1 is entered first, count remains 0 after the loop **B1**

(a) explains that this causes total / count to divide by zero, which is undefined/causes a runtime error, so no meaningful average can be produced **B1**

(a) Answer: If the user enters -1 as their very first input, the WHILE loop body never executes, so count stays at 0. The line average = total / count then attempts to divide by zero, which causes a runtime (division by zero) error rather than producing a sensible output.

(b) correct change to the loop condition so that any negative number (not just exactly -1) ends input, e.g. WHILE number ≥ 0 (oe) **B1**

(b) correct selection after the loop testing whether count == 0 **B1**

(b) correct output of a suitable message (e.g. "No valid numbers were entered") in the count == 0 branch **B1**

(b) correct calculation and output of average retained only in the ELSE branch (when count > 0), so division by zero can no longer occur **B1**

(b) Answer:

```
total = 0
count = 0
INPUT number
WHILE number >= 0
    total = total + number
    count = count + 1
    INPUT number
ENDWHILE
IF count == 0 THEN
    OUTPUT "No valid numbers were entered"
ELSE
    average = total / count
    OUTPUT average
ENDIF
```


© Revision Library | revisionlibrary.co.uk