CIE IGCSE Computer Science

Your notes

7.4 Validation and Testing

Contents

Validation


Your notes

Validation

- Validation and verification are used to ensure input data is correct, reasonable and accurate
- Validation generally is about making sure data follows a set of specified rules created by the programmer. Verification is about double-checking input data to make sure it's what the user intended to enter

Validation

- Validation is the method of checking input data so that it is of an expected value and therefore accepted by the system
- Programmers create rules using code that automatically checks user input data to make sure the data is reasonable. If it is not reasonable then the data is rejected, usually with a message explaining why it was rejected and allowing the user to reenter the data
- The different types of validation rules or checks are as follows:
  - Range checks
  - Length checks
  - Type checks
  - Presence checks
  - Format checks
  - Check digits
- Each piece of data may require multiple different checks to be fully valid

Range check

- Range checks make sure the input data is a value between a user-defined minimum and maximum value, for example, a percentage being between 0 and 100 inclusive or a date in April is between 1 and 30 inclusive

```
OUTPUT "Enter a number between 0 and 100"

REPEAT

    INPUT Number

    IF Number < 0 OR Number > 100

     THEN

        OUTPUT "Number is not between 0 and 100, please try again"

    ENDIF
```

```
UNTIL Number >= 0 AND Number <= 100
```


Your notes

Length check

- Length checks check either that the input data is of an exact number of characters or in a user specified number range of characters
- A bank 4-digit PIN number may be an example of an exact number of characters. If it is not 4 digits in length it should be rejected

```
OUTPUT "Please enter your 4 digit bank PIN number"

REPEAT

    INPUT Pin

    IF LENGTH(Pin) <> 4

      THEN

        OUTPUT "Your pin number must be four characters in length, please try again"

    ENDIF

UNTIL LENGTH(Pin) = 4
```

- Passwords usually have a specified range, for example, eight to twenty characters in length. If it does not fall within this range, it should be rejected

```
OUTPUT "Please enter a password between 8 and 20 characters"

REPEAT

    INPUT Password

    IF LENGTH(Password) < 8 OR LENGTH(Password) > 20

      THEN

        OUTPUT "Your password must be between 8 and 20 characters in length, please try again"

    ENDIF

UNTIL LENGTH(Password) >= 8 AND LENGTH(Password) <= 20
```

Type checks

- Type checks make sure the input data is of the correct data type. For example, someone's age should be an integer (a whole number) whereas their name should be a string (a bunch of characters)

```
OUTPUT "Enter an integer number"
```

```
REPEAT
    INPUT Number
    IF Number <> DIV(Number, 1)
      THEN
        OUTPUT "Not a whole number, please try again"
    ENDIF
UNTIL Number = DIV(Number , 1)
```

Presence check

- Presence checks make sure that input data has been entered and that the input box has not been left empty
- A login system requires presence checks as both a username and password are required for authentication

```
OUTPUT "Enter your username"
REPEAT
    INPUT Username
    IF Username = ""
      THEN
        OUTPUT "No username entered, please try again"
    ENDIF
UNTIL Username <> ""
```

Format check

- Format checks make sure that input data is of a predefined pattern
- Identification codes e.g. AP1234 and dates are examples of patterns
- Format checks are done using pattern matching and string handling
- The algorithm below checks a six digit identification number against the format "XX9999" where X is an uppercase alphabetical letter and 9999 is a four digit number
- The first two characters are checked against a list of approved characters. The first character is compared one at a time to each valid character in the ValidChars array. If it finds a match it stops


Your notes

looping and sets ValidChar to True. The second character is then compared one at a time to each valid character in the ValidChars array. If it finds a match then it also stops looping and sets ValidChar to True
- Casting is used on the digits to turn the digit characters into numbers. Once the digits are considered a proper integer they can be checked to see if they are in the appropriate range of 0-9999
- If any of these checks fail then an appropriate message is output


Your notes

```
INPUT IDNumber

IF LENGTH(IDNumber) <> 6

 THEN

        OUTPUT "ID number must be 6 characters long"

END IF

ValidChars ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

FirstChar ← SUBSTRING(IDNumber, 1, 1)

ValidChar ← False

Index ← 1

WHILE Index <= LENGTH(ValidChars) AND ValidChar = False DO

        IF FirstChar = ValidChars[Index]

         THEN

                ValidChar ← True

        ENDIF

        Index ← Index + 1

ENDWHILE

IF ValidChar = False

 THEN

        OUTPUT "First character is not a valid uppercase alphabetical character"

ENDIF

SecondChar ← SUBSTRING(IDNumber, 2, 2)

ValidChar ← False
```

```
Index ← 1

WHILE Index <= LENGTH(ValidChars) AND ValidChar = False DO

        IF SecondChar = ValidChars[Index]

         THEN

                ValidChar ← True

        ENDIF

        Index ← Index + 1

ENDWHILE

IF ValidChar = False

 THEN

        OUTPUT "Second character is not a valid uppercase alphabetical character"

ENDIF

Digits ← INT(SUBSTRING(IDNumber, 3, 6))

IF Digits < 0000 OR Digits > 9999

 THEN

        OUTPUT "Digits invalid. Enter four valid digits in the range 0000-9999"

ENDIF
```

Your notes

Check digits

- Check digits are numerical values that are the final digit of a larger code such as a barcode or an International Standard Book Number (ISBN). They are calculated by applying an algorithm to the code and are then attached to the overall code
- Check digits help to identify errors in data entry such as mistyping, miscanning or misspeaking
  - Such errors include missing or additional digits, swapped digits, mispronunciation of digits or simply an incorrect digit

```
Barcode ← "9780201379624"

Total ← 0

FOR Index = 1 to LENGTH(Barcode) - 1
```

```
        IF Index MOD 2 = 0
          THEN
                Total ← Total + CAST_TO_INT(Barcode[Index])*3
        ELSE
                Total ← Total + CAST_TO_INT(Barcode[Index])*1
        ENDIF
NEXT Index
CheckDigit ← 10 - Total MOD 10

IF CheckDigit = Barcode[LENGTH(Barcode)]
 THEN
        OUTPUT "Valid check digit"
ELSE
        OUTPUT "Invalid check digit"
ENDIF
```


Your notes

Verification


Your notes

- Verification is the act of checking data is accurate when entered into a system
- Mistakes such as creating a new account and entering a password incorrectly mean being locked out of the account immediately
- Verification methods include: double entry checking and visual checks

Double entry checking

- Double entry checking involves entering the data twice in separate input boxes and then comparing the data to ensure they both match. If they do not, an error message is shown

```
REPEAT
    OUTPUT "Enter your password"
    INPUT Password
    OUTPUT "Please confirm your password"
    INPUT ConfirmPassword
    IF Password <> ConfirmPassword
      THEN
        OUTPUT "Passwords do not match, please try again"
    ENDIF
UNTIL Password = ConfirmPassword
```

Visual check

- Visual checks involve the user visually checking the data on the screen. A popup or message then asks if the data is correct before proceeding. If it isn't the user then enters the data again

```
REPEAT
    OUTPUT "Enter your name"
    INPUT Name
    OUTPUT "Your name is: ", Name, ". Is this correct? (y/n)"
    INPUT Answer
UNTIL Answer = "y"
```

Worked example

Describe the purpose of validation and verification checks during data entry. Include an example for each.

[4]

Validation check

[1] for description:

- To test if the data entered is possible / reasonable / sensible
- A range check tests that data entered fits within specified values

[1] for example:

- Range / length / type / presence / format

Verification check

[1] for description:

- To test if the data input is the same as the data that was intended to be input
- A double entry check expects each item of data to be entered twice and compares both entries to check they are the same

[1] for example:

- Visual / double entry

Your notes

Test Data


Your notes

Test Data

Suggesting and applying suitable test data

- Before a system is used, each sub-system must be tested to ensure it works correctly and interacts correctly with other sub-systems
- Programs are tested by running them on a computing device while pseudocode and flowcharts must be dry run manually. Both require suitably chosen and different sets of test data. The outputs are then compared to the expected output to check if the algorithm works as intended
- Test data comes in several generic types:
  - **Normal**
    - Normal test data is data that a system would be expected to handle on a day-to-day basis, be accepted by the algorithm and produce expected results
    - Examples could include entering people's names and addresses, phone numbers, student grades as a percentage, etc
    - Student percentage grades could involve test data such as: 34, 41, 56, 78, 12, 92
  - **Abnormal**
    - Also known as erroneous data, abnormal data is data that is expected to fail and should be rejected by the system. This is used to prove that the system works correctly by rejecting incorrect data
    - Examples of abnormal data would be entering numbers instead of someone's name, or entering text instead of numbers
    - Student percentage grades abnormal data could involve test data such as: abc, 7&n, Harry, £300, <!%, etc
  - **Extreme**
    - Extreme test data is the maximum and minimum values of normal data that are accepted by the system
    - Examples could include percentages (0 and 100), days in April (1 and 30) or the number of characters in a passwords range
      - For an 8–20 character range password, a password of 8 characters and another of 20 characters would be tested
  - **Boundary**
    - Boundary test data is similar to extreme data except that the values on either side of the maximum and minimum values are tested
    - The largest and smallest acceptable value is tested as well as the largest and smallest unacceptable value
    - For example, a percentage boundary test would be 0 and -1 (for 0) and 100 and 101 (for 100). For the days in April, 1 and 0 (for day 1) and 30 and 31 (for day 30) would be tested. For an 8–20

length password, an 8 and 7 character password would be tested as well as a 20 and 21 character password

Your notes

Worked example

A programmer has written an algorithm to check that prices are less than $10.00

These values are used as test data: 10.00 9.99 ten

State why each value was chosen as test data.

[3]

10.00 is boundary or abnormal data and should be rejected as it is out of range [1]

9.99 is boundary, extreme and normal data and should be accepted as it is within the normal range [1]

Ten is abnormal data and should be rejected as it is the wrong value type [1]

Trace Tables


Your notes

Trace Tables

- Trace tables are used to follow algorithms and make sure they perform the required task correctly. Test data is usually used in conjunction with a trace table to ensure the correctness of the algorithm. Manually tracing an algorithm with test data in this way is known as a dry run
- Trace tables can be used with flowcharts or pseudocode or even real code if necessary
- Trace tables can also be used to discover the purpose of an algorithm by showing output data and intermediary steps
- Trace tables record the state of the algorithm at each step or iteration. The state includes all variables that impact the algorithms output as well as the output itself
- A trace table is composed of columns where each variable and the output is a column
- Whenever a value changes or an output is produced the relevant column and row is updated to reflect the change

Trace Table Walkthrough

- Below is a flowchart to determine the highest number of ten user entered numbers
- The algorithm prompts the user to enter the first number which automatically becomes the highest number entered
- The user is then prompted to enter nine more numbers. If a new number is higher than an older number then it is replaced
- Once all ten numbers are entered, the algorithm outputs which number was the highest
- Example test data to be used is: 4, 3, 7, 1, 8, 3, 6, 9, 12, 10

Your notes

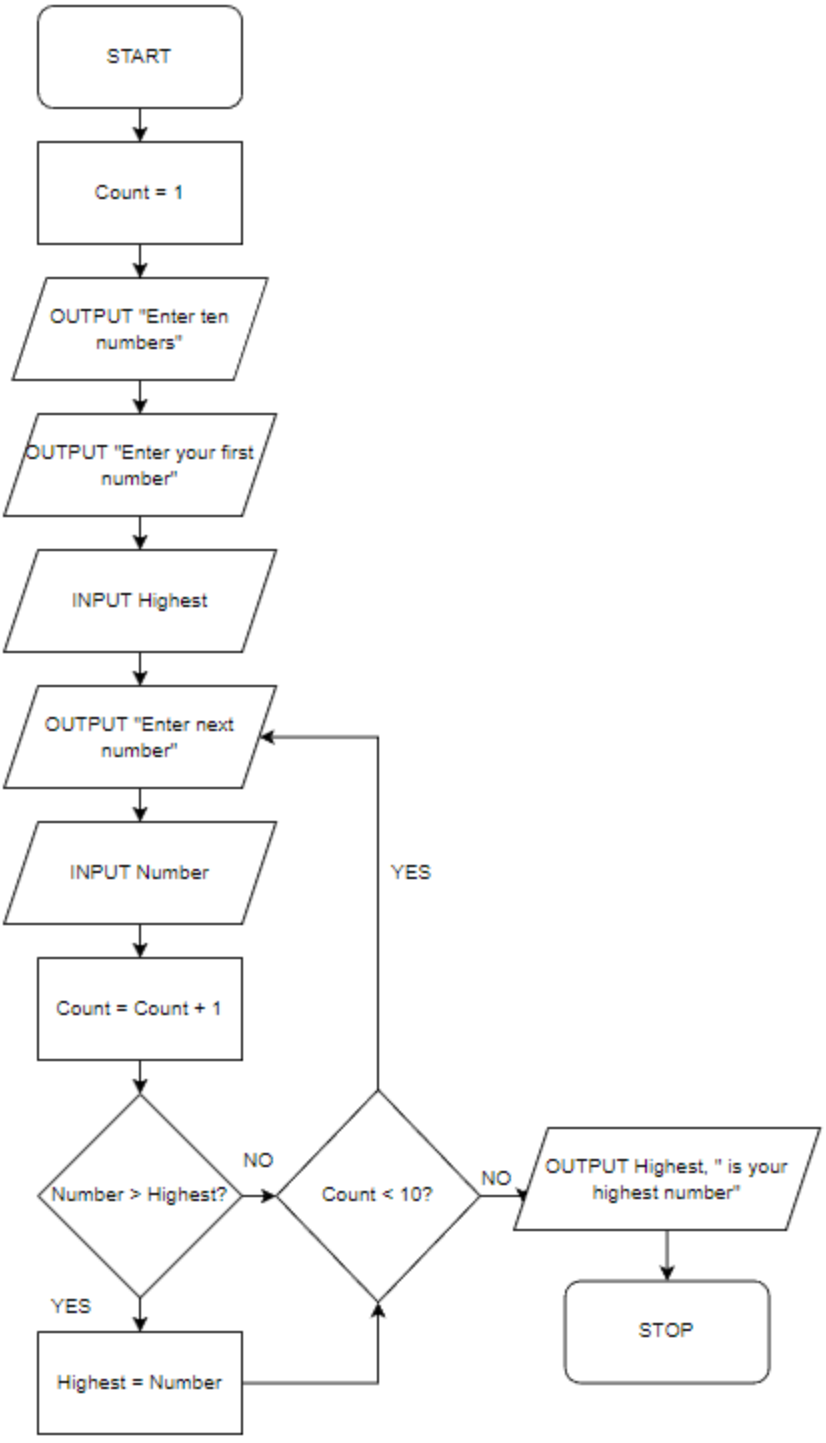


Figure 1: A flowchart to determine the highest of ten user entered numbers

Your notes

Trace table for Figure 1: Highest number			
Count	Highest	Number	Output
1			Enter ten numbers
	4		Enter your first number
2		3	Enter your next number
3	7	7	
4		1	
5	8	8	
6		3	
7		6	
8	9	9	
9	12	12	
10		10	12 is your highest number

Exam Tip

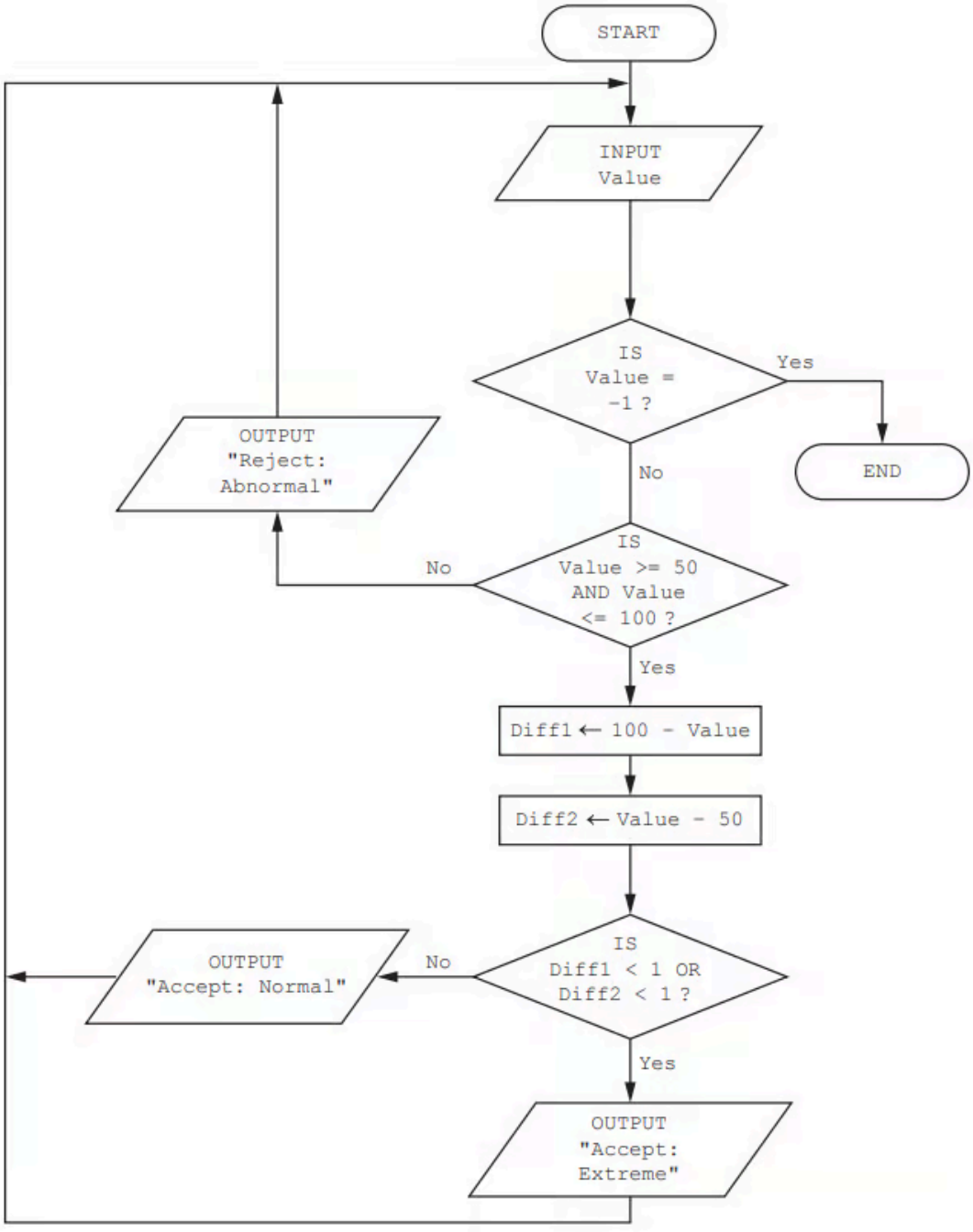
- When asked to identify the purpose of an algorithm in the exam, the variables listed will often be a single letter rather than a meaningful identifier

Worked example

The flowchart represents an algorithm. The algorithm will terminate if -1 is entered.

Your notes

Your notes



Complete the trace table for the input data: 50, 75, 99, 28, 82, 150, -1, 672, 80

[4]

Your notes

Value	Diff1	Diff2	Output

[1] for each correct column

Value	Diff1	Diff2	Output
50	50	0	Accept: Extreme
75	25	25	Accept: Normal
99	1	49	Accept: Normal
28			Reject: Abnormal
82	18	32	Accept: Normal
150			Reject: Abnormal
-1			