Chapter 12 Student Book Answers

12.1 What you should already know

Stages of the systems development lifecycle

- Analysis
- Design
- Coding
- Testing
- Maintenance

Types of program development lifecycle

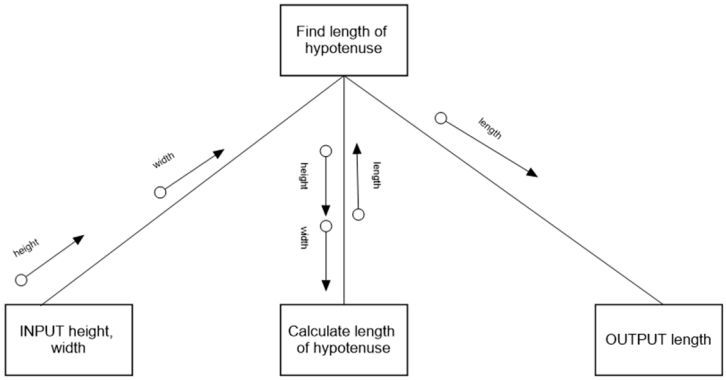
- Waterfall
- V-shaped model
- Iterative model
- Prototyping

12.2 What you should already know

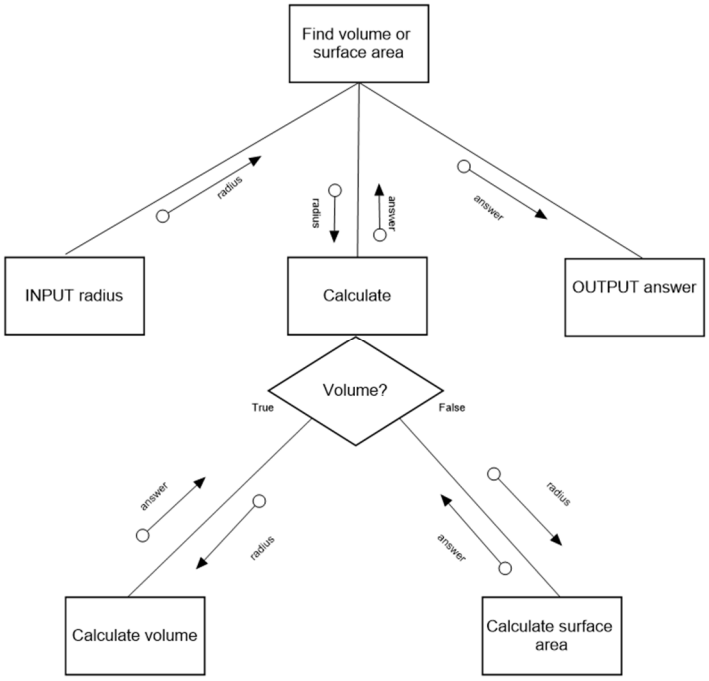
Example answer

```
// procedure to convert Celsius to Fahrenheit
DECLARE myTemp : REAL
PROCEDURE Fahrenheit (BYREF temperature : REAL)
    temperature ← temperature * 9 / 5 +32
ENDPROCEDURE

CALL Fahrenheit (myTemp)


// function to convert Celsius to Fahrenheit
DECLARE myTemp : REAL
FUNCTION fahrenheit (temperature : REAL)RETURNS REAL
    temperature ← temperature * 9 / 5 +32
ENDFUNCTION

myTemp ← fahrenheit (myTemp)
```

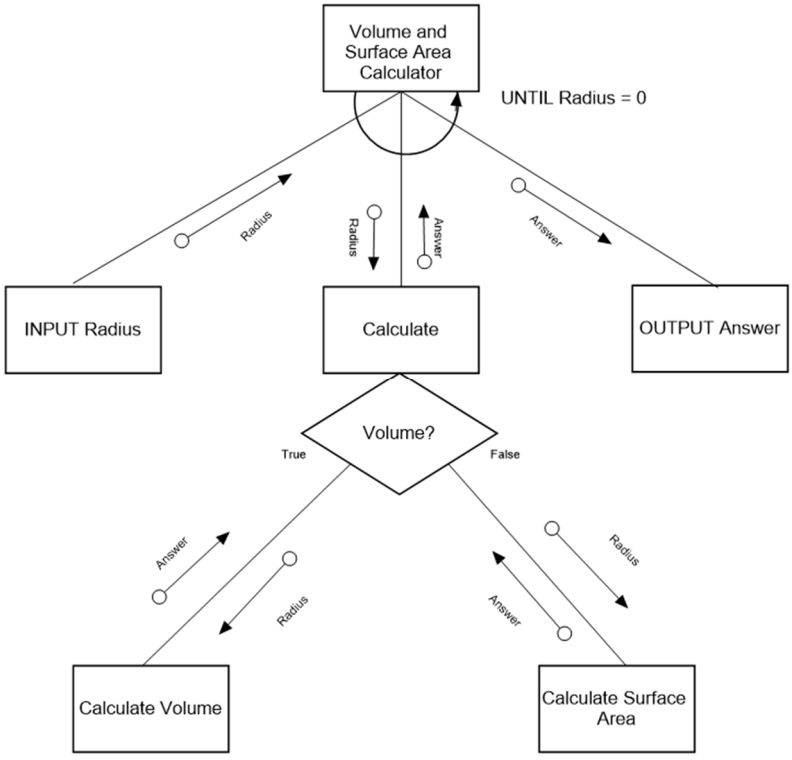
Activity 12A



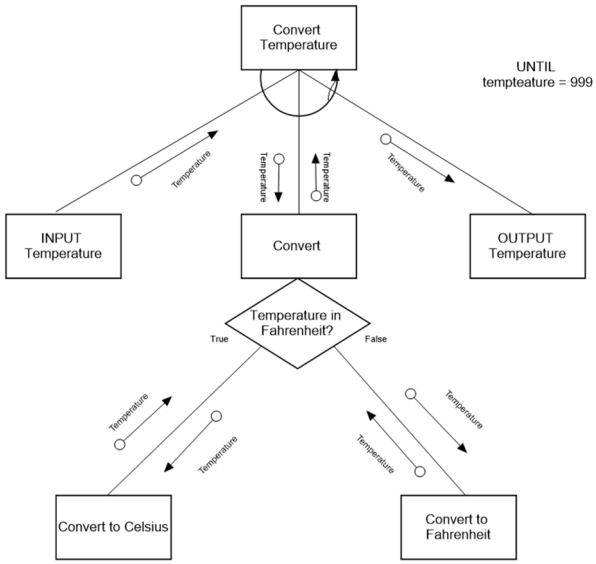
Activity 12B



Activity 12C



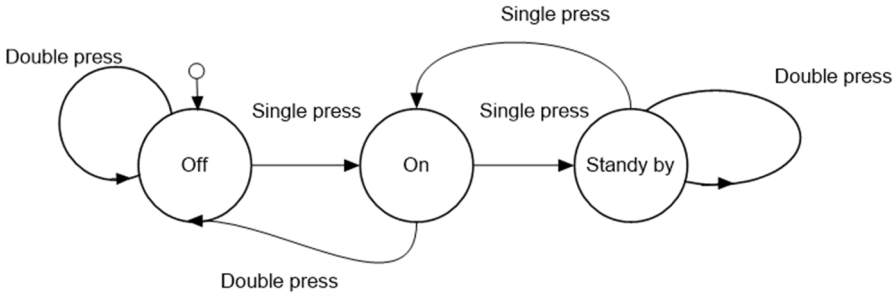
Activity 12D




```
// function to convert Celsius to
Fahrenheit
DECLARE myTemp : REAL
DECLARE myScale : CHAR
DECLARE finish : BOOLEAN

Finish ← FALSE

// function to convert Celsius to
Fahrenheit
FUNCTION fahrenheit (temperature :
REAL)RETURNS REAL
    temperature ← temperature * 9 / 5 +32
ENDFUNCTION

// function to convert Fahrenheit to Celsius
FUNCTION celsius (temperature : REAL)RETURNS REAL
    temperature ← (temperature – 32) * 5 / 9
ENDFUNCTION

REPEAT
    OUTPUT "Please enter value for temperature for conversion "
    INPUT myTemp
    OUTPUT "Please enter value C to convert to Celsius or F to convert to
            Fahrenheit "
    INPUT myScale
    IF myTemp <> 999
      THEN
        CASE OF myScale
          'C' : myTemp ← fahrenheit (myTemp)
          'F' : myTemp ← Celsius (myTemp)
        ENDCASE
        OUTPUT " Temperature is ",myTemp
      ELSE
        finish ← TRUE
    ENDIF
UNTIL finish
```

Activity 12E

Current state	Event	Next state
Off	Single press	On
Off	Double press	Off
Standby	Single press	On
Standby	Double press	Standby
On	Single press	Standby
On	Double press	Off



Activity 12F

```
PROCEDURE calculation(number1, number2, sign)
  IF number2 <> 0
    THEN
      CASE sign OF
        '+' : answer $ number1 + number2
        '-' : answer $ number1 - number2
        '*' : answer $ number1 * number2
       '/' : answer $ number1 / number2
       OTHERWISE answer $ 0
      ENDCASE
    ELSE
      answer $ 0
  ENDIF
IF answer <> 0
  THEN
    OUTPUT answer
ENDIF
ENDPROCEDURE
```

number1	number2	sign	answer	OUTPUT
20	10	+	30	30
20	10	-	10	10
20	10	*	200	200
20	10	/	2	2
20	10	?	0	
20	0	/	0	

Activity 12G

Example test data

10, V 10, S 10, v 10, p ten, V −1, V

radius	reply	answer	OUTPUT
10	V	30	Volume 4186.7
10	S	10	Surface Area 1236.8
10	v	200	Surface Area 1236.8
10	p	2	Surface Area 1236.8
ten	V	0	Input error
-1	V	0	Please enter a positive number

Problems:

- S is not checked for
- No checking that the input is numeric

Activity 12H

Example test data

Boundary for length

Lower bound Rejected `Abcdef1` Accepted `Abcdefg1`

Upper bound Rejected `Abcdefghijklmno1` Accepted `Abcdefghijklmn1`

Format

Rejected `Abcdefgh, Abcdefg!` Accepted `Abcdefg1`

Problems:

- Multiple ways of rejection – need to devise test data that fails both the format and the length for example `abcdefghijklmno1`

Activity 12I

Sample improvements:

1. Allow for repeated values to be entered.
2. Allow either upper or lower case ‘S’ or ‘V’ to be entered.
3. Reject all other values.
4. Output answers correct to 2 decimal places etc.

Activity 12J

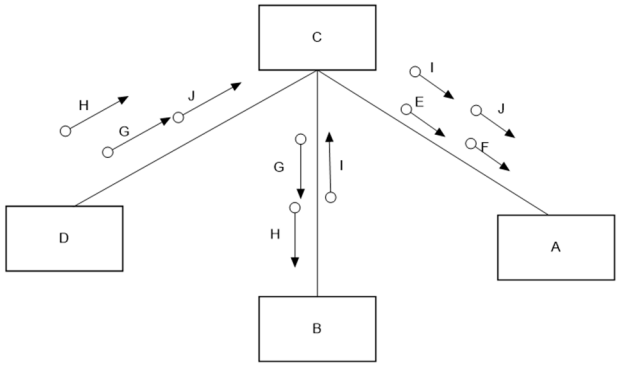
1 Syntax error in the grammar of the source code, e.g. `prnt` instead of `print`.
Logic error in the solution provided by the program, e.g. an incorrect test for values of 60 and over `IF value > 60  THEN` … does not include 60.

2 Normal data 50 should be accepted
Erroneous data 120 should be rejected
Boundary data (for integer values) 10 should be rejected, 11 should be accepted.

3 See12.3.4

End of chapter questions

1 a) b)



2

Line number	Error	Correction
01	Wrong procedure name – `"SortArray"`	`PROCEDURE ArraySort`
02	Wrong data type - `CHAR`	`DECLARE Temp: STRING`
03	Variables undefined	`DECLARE FirstID, SecondID, I, J : INTEGER`
04	Wrong `'Value2'` of 100	`FOR I ← 1 TO 99`
05	Wrong range	`FOR J ← 1 TO (100 – I)`
06/07	Wrong function - `MODULUS`	Replace `MODULUS` with `TONUM`: `FirstID ← TONUM(LEFT(Product[J],`
06/07	Wrong value of 6	Should be 4:
10	Assigning wrong value to `Temp`	`Temp ← Product[J]`
11	Assigning wrong value to `Product[I]`	`Product[J] ← Product[J + 1]`
13/14	Lines reversed	`13 ENDIF` `14 ENDFOR`

3 a) 1D-array of `INTEGER`.

b) i)

x	DayNumber	OUTPUT
0	1	
	2	
1	3	5/6/2015
	4	
2	5	7/6/2015
	6	
3	7	9/6/2015
		3

ii) The algorithm loops through days 1 to 7, adds the sales and, if they are greater than 10, adds 1 to x and outputs the day of those sales. At the end it outputs x, the total number of days where sales are greater than 10