INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

1. Algorithmic Thinking and Design



https://introcs.cs.rutgers.edu



INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

1. Algorithmic Thinking

- What is Algorithmic thinking?
- Algorithm development
- A Pseudo Language
- From Algorithm to pseudocode

Styles Adopted from Slides by Sedgewick and Wayne

LEARN HOW TO TELL a computer what to do

Machine language

- Easy for computer.
- Error-prone for human.

```
10: 8A00   RA ← mem[00]
11: 8B01   RB ← mem[01]
12: 1CAB   RC ← RA + RB
13: 9C02   mem[02] ← RC
14: 0000   halt
```

Adding two numbers

Natural language

- Easy for human.
- Error-prone for computer.



Actual newspaper headlines

—Rich Pattis

High-level language

- Some difficulty for both.
- An acceptable tradeoff.

```
for (int t = 0; t < 2000; t++){
   a[0] = a[11] * a[9];
   System.out.print(a[0]);
   for (int i = 11; i > 0; i--)
      a[i] = a[i-1];
}
```

But *which* high-level language?

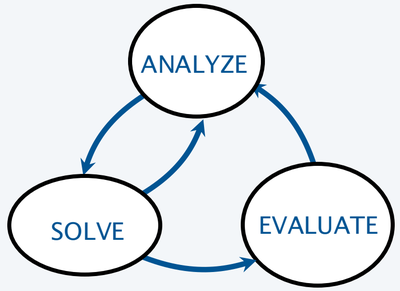


But first: Learn to think algorithmically.

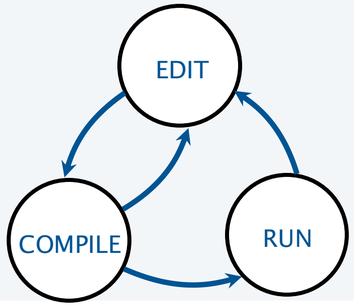
Algorithmic thinking (Solving problems using PSEUDOCODE)

Learn how to analyze and solve computational problems at a level of abstraction .









A significant difference with coding : *No implementation details required. NOT WORRIED ABOUT SYNTAX).*

We use set of pseudocode instructions to express a solution algorithmically. **THERE ARE RULES WE MUST FOLLOW!!!** ***See Pseudocode Reference Guide.***

An algorithm: A step-by-step process of solving a problem

Problem

- Find the sum of two numbers.

What we need: A vocabulary for describing an algorithm.

Algorithm

READ x

READ y

DISPLAY x + y



1. Algorithmic Thinking

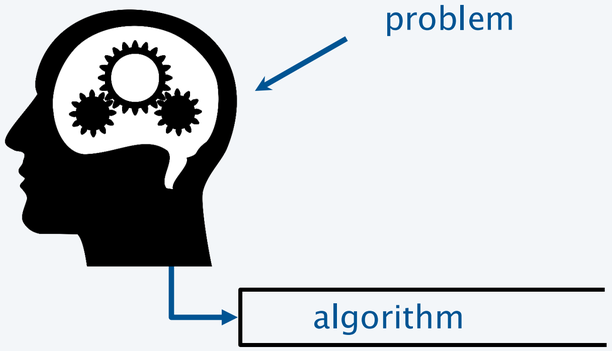
- What is Algorithmic thinking?
- Algorithm development
- A Pseudo Language
- From Algorithm to pseudocode

Computational thinking (CT) (**decomposition, pattern recognition, abstraction and algorithms)**

is a set of problem-solving methods that involves expressing problems and their solutions.

Solution can be translated to an algorithm

Convert the algorithm to a program that can be solved using a computer.



Problem ➔ CT ➔ Algorithm

Thinking Computationally

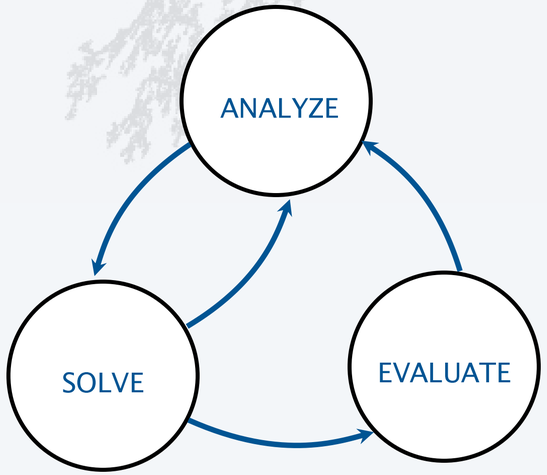
- Fundamental Skill for everyone, not just computer scientists.
- Integration of computational ideas in other disciplines.

Stages of CT

- Abstraction problem formulation.
- Automation expressing a solution that a machine can execute.
- Analyses solution execution and evaluation.

Coined by Jeannette M. Wing

1. Algorithmic Thinking



- What is Algorithmic thinking?
- Algorithm development
- A Pseudo Language
- From Algorithm to pseudocode

Pseudo Language

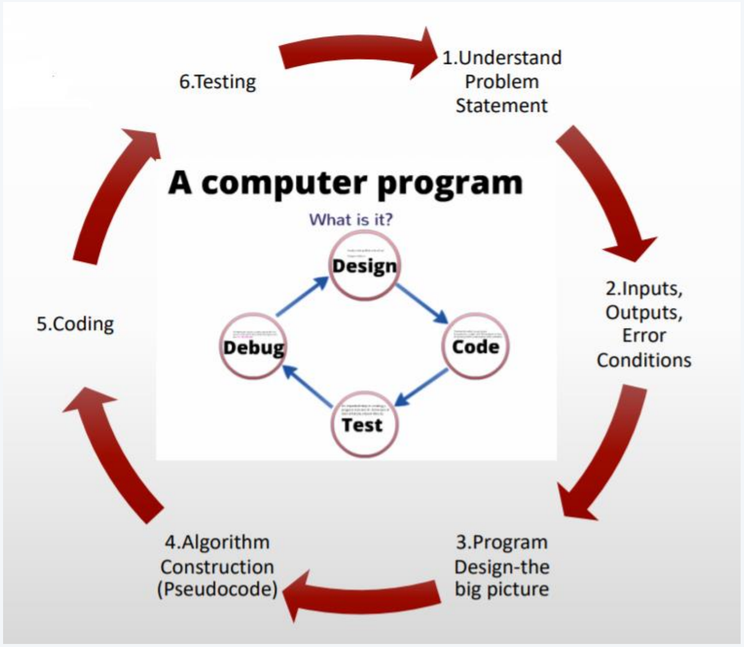
An informal high-level description of a computer program

Structure uses same structural conventions as a programming language.

Readability intended for human reading not machine reading.

Software Design Cycle

- Pseudo Language help express the algorithm in the design cycle.
- Coding help avoid expensive errors prior to coding.



Software Design Life Cycle

High-level components in Program Development

- Input how a program receives input data.
- output how a program returns the results.
- compute how a program performs its tasks.

Definitions – SEE PSEUDOCODE REFERENCE GUIDE

Define. Terms used in programming using a pseudo language

Definitions

<variable> an entity that may change its value (e.g. temperature)

<constant> a value that does not change (e.g. 4)

<expression> variables, constants and operators grouped together to show the value of something (e.g. x*5 + y)

<Boolean expression> an expression that evaluates to TRUE or FALSE (e.g. n > 3)

<statement> a combination of expressions (e.g. COMPUTE x AS x + 3)

A Pseudo language vocabulary

I/O
READ
DISPLAY

Initialization computation
SET
COMPUTE

Flow control
IF-THEN
IF-THEN-ELSE
NESTED IF-THEN-ELSE

Repetition
WHILE-ENDWHILE
DO-WHILE
REPEAT-UNTIL
FOR-ENDFOR

Nested
NESTED-FOR
NESTED-FOR-IF

Increment/decrement
ADD
SUBTRACT

Comparisons
<
<=
>
>=
is
!=

Arithmetic Operations
+
-
*
/
%

Boolean operations
NOT
AND
OR

Terminate Program
HALT

Your pseudocode will primarily consist of these constructs plus identifiers (names) that you make up. A *Pseudocode Reference Sheet* is posted.

I/O - READ and DISPLAY

Q. How does READ and DISPLAY work (in pseudo language)?

A. READ. Reads a value into a variable

DISPLAY. Outputs a variable or constant

READ <variable>

DISPLAY <variable/constant>

Syntactical form

READ temperature

DISPLAY temperature

An Example

INITIALIZE

Q. How do we initialize a variable (a placeholder) in a program?

A. Using the SET statement in the pseudo language.

SET <variable> **TO** <constant | variable>

Syntactical Form

SET x **TO** 10

SET x **TO** y

An Example

COMPUTE

Q. How do we compute an expression in a program?

A. Using the COMPUTE statement in the pseudo language.

COMPUTE <variable> **AS** <expression>

Syntactical Form

COMPUTE x **AS** 10 * 12 + 20

COMPUTE x **AS** y + z

An Example

Flow Control - One-way Branching

Q. How does one-way branching work?

A. IF-THEN. One-way branching based on the condition being TRUE

```
IF <Boolean-expression> THEN
    <statement/statements>
ENDIF
```

syntactical form

```
IF (number > 0) THEN
    DISPLAY positive
ENDIF
```

An Example

Flow Control - Two-way Branching

Q. How does two-way branching work?

A. IF-THEN-ELSE. Two-way branching is based on Condition being TRUE or FALSE

```
IF <Boolean-condition> THEN
    <statement/statements>
ELSE
     <statement/statements>
ENDIF
```

Syntactical form

```
IF temperature < 40 THEN
    DISPLAY cold day
ELSE
    DISPLAY mild day
ENDIF
```

An Example

Flow Control - Multi-way Branching

Q. How does multi-way branching works?

A. NESTED IF-THEN-ELSE. multi-way branching based on Condition being TRUE or FALSE and more

```
IF <Boolean-condition> THEN
    <statement/statements>
ELSE
   IF <Boolean-condition> THEN
        <statement/statements>
   ELSE
      <statement/statements>
   ENDIF
ENDIF
```

A syntactical form

```
IF temperature < 40 THEN
   DISPLAY cold day
ELSE
   IF temperature >= 80 THEN
        DISPLAY hot day
   ELSE
        DISPLAY mild day
   ENDIF
ENDIF
```

3-way branching example

ADD and SUBTRACT

Q. How do we increment or decrement a variable?

A. Using the ADD or SUBTRACT statement in the pseudo language.

ADD <variable/constant> **TO** <variable>

SUBTRACT <variable/constant> **FROM** <variable>

Syntactical Form

ADD 1 **TO** x

ADD y **TO** x

SUBTRACT 1 **FROM** x

SUBTRACT x **FROM** y

Example

Iterations - WHILE loop

Q. How does WHILE loop works?

A. Initialization. (optional)

Boolean Expression. Evaluate a Boolean expression

IF TRUE. Execute a sequence of statements

Repeat

```
<initialization statements>
WHILE <Boolean-expression>
    <statements if Boolean expression is true>
ENDWHILE
```

Syntactical Form

```
SET n TO 3
SET sum TO 0
WHILE n > 0
  COMPUTE sum AS sum + n
  SUBTRACT 1 FROM n
ENDWHILE
```

An example

PSEUDOCODE Comparison operators

Fundamental operations that are defined for each primitive type allow us to *compare* values.

- Operands: two expressions of the same type.
- Result: a value of type boolean.

operator	*meaning*	true	false
is	equal	2 is 2	2 is 3
!=	not equal	3 != 2	2 != 2
<	less than	2 < 13	2 < 2
<=	less than or equal	2 <= 2	3 <= 2
>	greater than	13 > 2	2 > 13
>=	greater than or equal	3 >= 2	2 >= 3



1. Algorithmic Thinking

- What is Algorithmic thinking?
- Algorithm development
- A Pseudo Language
- From Algorithm to pseudocode

Inputs, outputs and error conditions

LO 1.1

Problem. Ellen and Ana are moving together. Determine how many cats they will have in total?

Solution Design. Find out how many cats each person have.

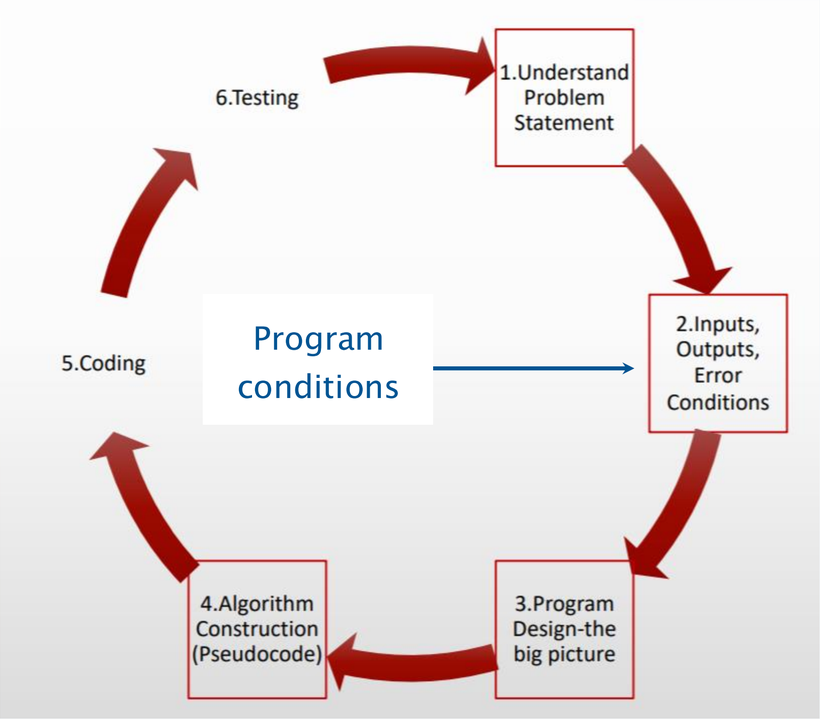
Compute and display the sum.

Identify.

Inputs. Ellen's number of cats, Ana's number of cats

Output. The total cats

Error conditions. Bad input (e.g. Ana's cats is -5)



Example Program 1 (*Think operations and test cases)*

Problem. Ellen and Ana are moving together. Determine how many cats they will have in total?

Solution Design. Find out how many cats each person has.

Compute and display the sum.

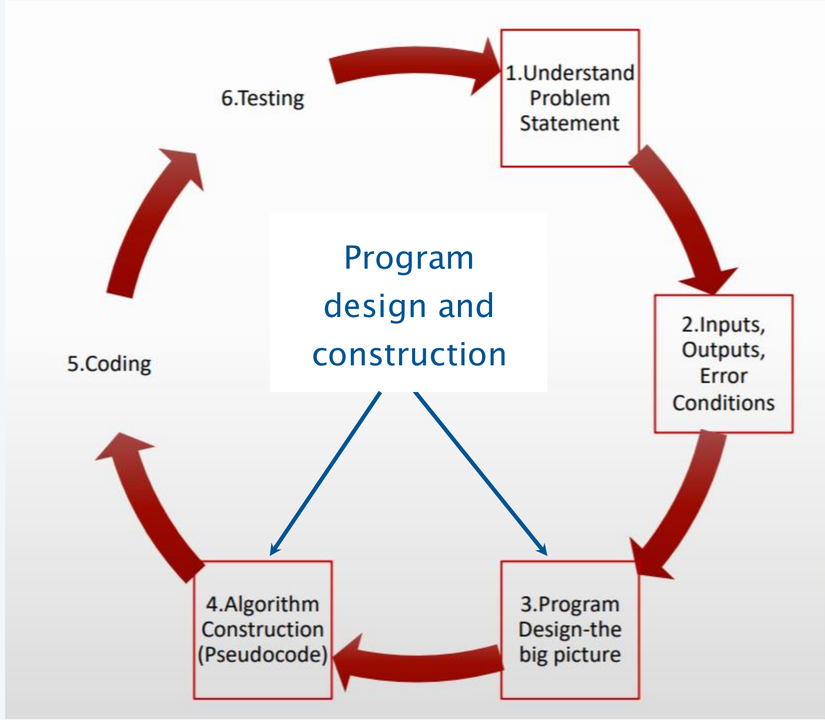
PSEUDOCODE:

READ EllensCats

READ AnasCats

COMPUTE sumOfCats **AS** EllensCats + AnasCats

DISPLAY sumOfCats



Example Program 1 (with error handling)

Problem. Ellen and Ana are moving together. Determine how many cats they will have in total?

Solution Design. Find out how many cats each person have.
Handle error in input
Compute and display the sum.

ALGORITHM:

```
READ EllensCats
READ AnasCats
IF EllensCats < 0 OR AnasCats < 0 THEN
    DISPLAY ERROR
ELSE
    COMPUTE sumOfCats AS EllensCats + AnasCats
    DISPLAY sumOfCats
ENDIF
```

Test Cases

[input][output] pairs

[1, 3][4]

[0, 2][2]

[2, -5][ERROR]

[-3, 2][ERROR]

[-3, -4][ERROR]

Example Program 2 –START HERE

Problem. Calculate salary based on hours worked and hourly rate.

Solution Design. Input: hoursWorked and ratePerHour

Output: totalPay

Error: negative inputs

ALGORITHM:

```
READ hoursWorked
READ ratePerHour
IF hoursWorked < 0 OR ratePerHour < 0 THEN
    DISPLAY ERROR
ELSE
    COMPUTE totalPay AS hoursWorked * ratePerHour
    DISPLAY totalPay
ENDIF
```

Test Cases

[input][output] pairs

[8, 10][80]

[0, 20][0]

[2, 0][0]

[-3, 20][ERROR]

[10, -20][ERROR]

Example Program 3

Problem. Determine if an integer is even or odd.

Solution Design. Input: an integer

Output: even or odd

Error: none

Precondition: input is an integer

ALGORITHM:

READ number	← 1 operation
IF number % 2 is 0 **THEN**	← 1 operation
DISPLAY even	← 1 operation
ELSE	OR
DISPLAY odd	← 1 operation
ENDIF	Total of 3 operations

Test Cases

[input][output] pairs

[-2][even]

[0][even]

[2][even]

[3][odd]

Example Program 4

```
SET n TO 3
SET f TO 0
SET g TO 1
SET i TO 0
WHILE i <= n
    DISPLAY f
    COMPUTE f AS f + g
    COMPUTE g AS f – g
    COMPUTE i AS i + 1
ENDWHILE
```

WHAT IS PRINTED?

Example Program 5

Problem. A cake up to 6” cost $10, 6”-12” cost $15, greater than 12” cost $20. Given the diameter of the cake, find the cost.

Solution Design. Input: diameter of the cake (integer)

Output: the price

Error: diameters less than or equal to 0

Example Program 5 continued


```
READ cakeSize
  IF cakeSize <= 0 THEN
     DISPLAY error
  ELSE
     IF cakeSize < 6 THEN
        SET price TO 10
     ELSE
        IF cakeSize >= 6 AND cakeSize <=12 THEN
           SET price TO 15
        ELSE
           SET price TO 20
        ENDIF
     ENDIF
     DISPLAY price
  ENDIF
```

Test Cases:

[input][output] pairs

[-3][error]

[0][error]

[5][10]

[6][15] *boundary condition*

[8][15]

[12][15] *boundary condition*

[13][20]

[15][20]

Example Program 5 continued


```
READ cakeSize
  IF cakeSize <= 0 THEN
     DISPLAY error
     HALT
  ENDIF
  IF cakeSize < 6 THEN
        SET price TO 10
  ENDIF
  IF cakeSize >= 6 AND cakeSize <=12 THEN
        SET price TO 15
  ENDIF
  IF cakeSize > 12 THEN
        SET price TO 20
  ENDIF
  DISPLAY price
```

HALT added

Test Cases:

[input][output] pairs

[-3][error]

[0][error]

[5][10]

[6][15] *boundary condition*

[8][15]

[12][15] *boundary condition*

[13][20]

[15][20]

Example 5 continued: Compare - Contrast

Version 1

```
READ cakeSize
  IF cakeSize <= 0 THEN
     DISPLAY error
  ELSE
     IF cakeSize < 6 THEN
        SET price TO 10
     ELSE
        IF cakeSize >= 6 AND cakeSize <=12 THEN
           SET price TO 15
        ELSE
           SET price TO 20
        ENDIF
     ENDIF
     DISPLAY price
  ENDIF
```

Version 2

```
READ cakeSize
  IF cakeSize <= 0 THEN
     DISPLAY error
     HALT
  ENDIF
  IF cakeSize < 6 THEN
        SET price TO 10
  ENDIF
  IF cakeSize >= 6 AND cakeSize <=12 THEN
        SET price TO 15
  ENDIF
  IF cakeSize > 12 THEN
         SET price TO 20
  ENDIF
  DISPLAY price
```

Example Program 6

Problem. Convert Fahrenheit to Celsius.

Solution Design. Input: a temperature in Fahrenheit

Output: equivalent in Celsius

Error: temperature below absolute zero (-459.67F)

ALGORITHM:

READ tempF

IF tempF < -459.67 **THEN**

DISPLAY ERROR: Temperature must be greater than -459.67

ELSE

COMPUTE tempC **AS** (tempF – 32) * 5/9

DISPLAY tempC

ENDIF

Test Cases

[input][output] pairs

[32][0]

[-459.68][error]

[-459.67][273.15]

[65][18.33]

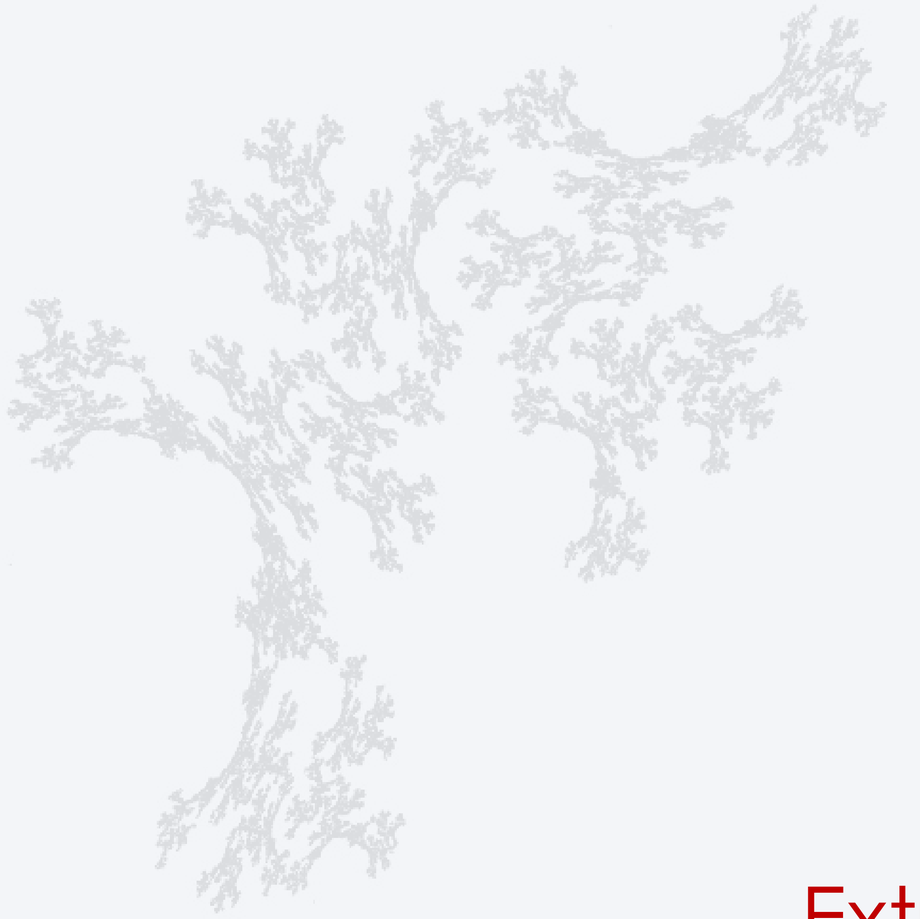
INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

1. Algorithmic Thinking and Design

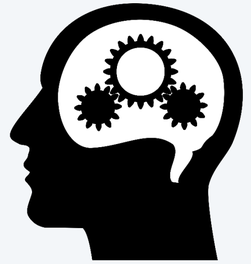


https://introcs.cs.rutgers.edu



INTRODUCTION TO
COMPUTER SCIENCE
Rutgers University

Extra Problems



Quiz-1

Identify each of the following as variable, arithmetic expression, Boolean expression or statement.

1. (x > 2)
2. 2*temperature
3. myGrade
4. anaHouse
5. (temp < 0) or (temp > 10)
6. COMPUTE x AS x - 3

Quiz-2

Write an algorithm, using pseudocode, to solve the following problems. Identify input, output and error conditions.

1. Display “Hello world”
2. Add two numbers
3. Add two positive numbers
4. Find the larger of two numbers
5. Find the larger of three negative numbers
6. Find the area of a circle given the radius
7. Print all even numbers from 1 to 100

Quiz3

A family visits the County Fair.

Admission to the County Fair is charged as follows:

- Free for children under 5 years old.
- $5 per person for adults 65 years old or older.
- $20 per person for all others.

Note: ages are > 0 and <= 120

- On Weekends there is a parking fee of $25 but on other days parking is free.
  - Assume value entered for this input is either yes (it's a weekend) or no (it's not a weekend).
- Calculate and display the total cost for the family to attend the fair?

QUESTIONS

1. Give inputs, outputs, error conditions
2. List test cases [input, output] pairs
3. Write the pseudocode solution

Quiz-1 - SOLUTIONS

Identify each of the following as variable, arithmetic expression, Boolean expression or statement.

1. (x > 2) Boolean expression
2. 2*temperature arithmetic expression
3. myGrade variable
4. anaHouse variable
5. (temp < 0) or (temp > 10) Boolean expression
6. COMPUTE x AS x – 3 statement

Pseudocode – solution #1

- **Display “hello world”**

```
DISPLAY “hello world”
```

- **Add two numbers**

```
READ num1
READ num2
COMPUTE sum AS num1 + num2
DISPLAY sum
```

Pseudocode – solution #2, #3

- **Add two positive numbers**

```
READ num1
READ num2
IF (num1 <= 0) OR (num2 <= 0) THEN
    DISPLAY “ERROR”
ELSE
    COMPUTE sum AS num1 + num2
    DISPLAY sum
ENDIF
```

5 or 6 operations

Pseudocode solution #4

- **Find (and display) the larger of two numbers**

```
READ num1
READ num2
IF num1 > num2 THEN
   DISPLAY num1
ELSE
   DISPLAY num2
ENDIF
```

Pseudocode – solution #5

- **Find (and display) the largest of three negative numbers**

```
READ num1
READ num2
READ num3
IF (num1 >= 0) OR (num2 >= 0) OR (num3 >= 0) THEN
        DISPLAY ERROR
ELSE
        SET largest TO num1
        IF (num2 > largest) THEN
            SET largest TO num2
        ENDIF
        IF (num3 > largest) THEN
            SET largest TO num3
        ENDIF
        DISPLAY largest
ENDIF
```

Pseudocode – solution #5

- **Find (and display) the largest of three numbers—ANOTHER ALGORITHM**

```
READ num1
READ num2
READ num3
IF num1 <  num2 THEN
        IF num2 < num3 THEN
            DISPLAY num3
        ELSE
            DISPLAY num2
        ENDIF
ELSE
        IF num1 < num3 THEN
            DISPLAY num3
        ELSE
            DISPLAY num1
        ENDIF
ENDIF
```

Pseudocode – solution #6

- **Find (and display) the area of a circle given the radius**

```
READ radius
IF (radius <= 0) THEN
        DISPLAY ERROR
ELSE
        COMPUTE area AS 3.1415 * radius * radius
        DISPLAY area
ENDIF
```

Pseudocode – solution #7

Display all even positive integers less than or equal to 100

```
SET num TO 2
WHILE (num <= 100)
    DISPLAY num
    ADD 2 TO num
END WHILE
```

```
SET num TO 1
WHILE (num <= 100)
    IF num % 2 = 0 THEN
            DISPLAY num
     ENDIF
     ADD 1 TO num
END WHILE
```

Pseudocode – solution #8

INPUTS
- Number of people (numPeople)
- Age of each person (age)
- Weekend or not (weekend)

OUTPUT
- The total cost for family

ERROR CONDITIONS
- numPeople <=0
- age <=0
- age >120
- *No check for errors for weekend because the problem says ASSUME*

TEST CASES

[*numPeople ages of each person weekend or not,* ***TOTAL COST]***
[inputs][Output]

[3, 4, 30, 67, yes] [50]
[1, 65, no] [5]
[5, 3, 5, 40, 65, 70, yes] [75]
[-1][ERROR]
[3, -4, 30, 2][ERROR]

```
SET total TO 0
READ numPeople
IF numPeople <=0 THEN
        DISPLAY ERROR
        HALT
ENDIF
SET count TO 1
WHILE count <= numPeople
    READ age
    IF age <=0 OR age > 120 THEN
        DISPLAY ERROR
        HALT
    ENDIF
    IF age >= 5 AND age < 65
        ADD 20 TO total
    ENDIF
    IF age  >= 65 THEN
        ADD 5 TO total
    ENDIF
    ADD 1 TO count
ENDWHILE
READ weekend
IF weekend = YES THEN
    ADD 25 TO total
ENDIF
DISPLAY total
```