IB Computer Science (HL)
Paper 1: Study Guide

Table of Contents

Amith’s Pseudocode Guide

User Input

```
// Similar to print keyword in Python or Java
input S
input NAME
X = input(‘Enter the locked number’)
```

Displaying User Output

```
HOURS = 30
MINUTES = 5
output HOURS
output HOURS, “:”, MINUTES
output “Hello”
```

Declaring variables

```
MYWORD = “This is a word”
T, P // Declare 2 variables at once
T = 0; P = 0;
I = 55
boolean FOUND = False // Specifies data type
FOUND = True
```

Arrays

```
TEAM  // Array with 450 positions
TEAM[450] // Explicitly represents an initialized array with 450
positions, same as previous
GRADES ARRAY // Creates a new empty array called GRADES
WEEKS ARRAY  // Creates a new empty array called WEEKS
GRADES = [91, 77, 82] // Iin
X = TEAM.length // Returns the number of elements in an array
```

Collections (See Common Pseudocode Tasks for Examples)

`NAMES = {“John”, “Ali”, “Olha”}`	Declare a collection called `NAMES`
`NAMES.resetNext()`	Move pointer to beginning of collection (position is before first element)
`NAMES.hasNext()`	Checks whether next element in collection exists (Returns `True` if next element exists, otherwise `False`)
`NAMES.isEmpty()`	Checks whether collection is empty (Returns `True` if collection is empty, otherwise `False`)
`NAMES.getNext()`	Returns next element, moves current position to next element `X = NAMES.getNext()`
`NAMES.addItem(“Jabir”)`	Add item to collection

Arithmetic Operators

```
A = 3 + 3
B = 3 - 3
C = 3 * 3
D = 3 / 3
E = 3 mod 3
F = 3 div 3
G = abs(C-11) // Calculate the absolute value
```

Boolean Operators

```
if A > B then
endif

if A < B then
endif

if A >= B then
endif

if A <= B then
endif

if A != B then
endif

if A ≠ B then
endif

if not A = B then
endif

if !(A=B) then
endif

if A == B then
endif

if A = B then
endif

if A = B and C = D then
endif

if A = B or C = D then
endif
```

Conditionals (if Statements)

```
if condition then
      // Code
endif
```

```
if A > B then
    output A
endif
```

```
if condition then
      // Code
else
      // Code
endif
```

```
if A > B then
    output “A is greater than B”
else
    output “A is less than or equal”
endif
```

```
if condition then
      // Code
else if condition then
      // Code
else
      // Code
endif
```

```
if A > B then
     output “A is more than B”
else if A > C then
     output “A is more than C”
else
     output “A is less than both”
endif
```

While Loops

```
loop while condition
      // Code
endloop
```

```
I=0
loop while I < 5
    output I
    I=I + 1
endloop
```

For Loops

loop for *counter* from *start* to *end*
    // Code
endloop
	loop for I from 0 to 5
	    output I
	endloop

loop *counter* from *start* to *end*
    // Code
endloop
	loop I from 0 to 5
	    output I
	endloop

loop *counter*=*start* to *end* do
    // Code
endloop
	loop I=0 to 5 do
	     output I
	endloop

Functions (Methods)

method_name(*parameter_1*, *parameter_2*, …)
     // Code
     return *method_return_value*
end*method*
	sum(num_1, num_2)
	     total = num_1 + num_2
	     return total
	endsum

Queues (See Common Pseudocode Tasks for Examples)

`Q = new Queue()`	Declare a queue called `Q`
`Q.enqueue(3)`	Add an element to a queue (to the back of the queue)
`X = Q.dequeue()`	Remove an element from the queue (from the front of the queue)
`Q.isEmpty()`	Checks whether queue is empty (Returns `True` if queue is empty, otherwise `False`)

Stacks (See Common Pseudocode Tasks for Examples)

`S = new Stack()`	Declare a stack called `S`
`S.push()`	Add an element to a stack (to the top of the stack)
`S.pop()`	Remove an element from the stack (from the top of the stack)
`S.isEmpty()`	Checks whether the stack is empty (Returns `True` if stack is empty, otherwise `False`)

Pseudocode Common Tasks

Topic 4 Pseudocode

Iterate through an Array

```
// Prints out every element in TEAM
TEAM[450]
loop I from 0 to TEAM.length - 1
    output TEAM[I]
endloop
```

Iterate through a Collection

```
// Prints out every element in TEAM_COL
TEAM_COL = {...}  // This is a collection that is not empty
TEAM_COL.resetNext()
loop while TEAM_COL.hasNext()
    output TEAM.getNext()
endloop
```

Checking for Divisibility

```
// Check is the user input is divisible by 3
input NUM
if NUM mod 3 == 0
    output "the number you entered is divisible by 3."
else
    output "the number you entered is not divisible by 3."
endif
```

Array to Collection

```
NAMES = ["John", "Ali", "Olha"]
NAMES_COL = {}
loop I from 0 to NAMES.length-1
    NAMES_COL.addItem(NAMES[I])
endloop
```

Collection to Array

```
CLIENT_COL = {"Ella", "Jeff", "Samir"}
CLIENTS[3]
COUNTER = 0
CLIENT_COL.resetNext()
loop while CLIENT_COL.hasNext()
      CLIENTS[COUNTER] = CLIENT_COL.getNext()
      COUNTER = COUNTER + 1
endloop
```

Search Array (Range)

```
// Output all values in the SPEEDS array that are less than 40
SPEEDS = {71, 23, 45, 59, 92, 33, 65}
loop I from 0 to SPEEDS.length - 1
      if SPEEDS[I] < 40 then
            output SPEEDS[I]
      endif
endloop
```

Search Array (User Input)

```
// Output the index of the element in TEMPS that matches S, which is
user input
// If it's not found, print "Not found"
input S
TEMPS = [32, 45, 12, 34, 19, 25, 40, 39]
FOUND = False
loop I from 0 to TEMPS.length - 1
      if TEMPS[I] == S then
            output I
            FOUND = True
      endif
endloop
if FOUND == False then
      output "Not found"
endif
```

Search Collection (Range)

```
// Output all values in the collection greater than 50
GRADES = {71, 23, 45, 59, 92, 33, 65}
GRADES.resetNext()
loop while GRADES.hasNext()
     G = GRADES.getNext()
     if G > 50 then
          output G
     endif
endloop
```

Search Collection (User Input)

```
// If S is found in collection, output “found”, otherwise output “not
found”
input S
COL = {“Amith”, “James”, “Stephen”, “Nguyen”, “Sato”, “Fernando”}
FOUND = False
COL.resetNext()
loop while COL.hasNext()
     if S = COL.getNext() then
          output “Found”
          FOUND = True
     endif
endwhile
if FOUND = False then
     output “Not Found”
endif
```

Average in Array

```
// Print the average of all the elements in an integer
TEMPS = [71, 23, 45, 59, 92, 33, 65]
SUM = 0
loop I from 0 to TEMPS.length-1
     SUM = TEMPS[I] + SUM
AVG = SUM/TEMPS.length
output(AVG)
```

Max in Array

```
// Find the maximum value in an array of integers
TEMPS = [71, 23, 45, 59, 92, 33, 65]
MAX = TEMPS[0]
loop I from 0 to TEMPS.length-1
     if TEMPS[I] > MAX then
          MAX = TEMPS[I]
     endif
endloop
```

Min in Array

```
// Find the maximum value in an array of integers
TEMPS = [71, 23, 45, 59, 92, 33, 65]
MIN = TEMPS[0]
loop I from 0 to TEMPS.length-1
     if TEMPS[I] < MIN then
          MIN = TEMPS[I]
     endif
endloop
```

Validation

```
// Make sure that user input is positive and greater than 0
input S
loop while S < 1 then
     output "Enter a value greater than 0:"
     input S
endloop
```

Find values in common between two arrays

```
FAILED_MATH = ["James", "Pablo", "Alvaro", "Sean", "Zach",
"Fernando"]
FAILED_ENGLISH = ["Pablo", "Fernando", "Sara", "Alice"]
for I loop from 0 to FAILED_MATH.length-1
     for J loop from 0 to FAILED_ENGLISH.length-1
          if FAILED_MATH[I] == FAILED_ENGLISH[J] then
               output(FAILED_MATH[I])
          endif
     endloop
endloop
```

Find values in common between two collections

```
FAILED_MATH_COL = {“James”, “Pablo”, “Alvaro”, “Sean”, “Zach”,
“Fernando”}
FAILED_ENGLISH_COL = {“Pablo”, “Fernando”, “Sara”, “Alice”}
FAILED_MATH_COL.resetNext()
FAILED_ENGLISH_COL.resetNext()
loop while FAILED_MATH_COL.hasNext()
     MATH = FAILED_MATH_COL.getNext()
     FAILED_ENGLISH_COL.resetNext()
     loop while FAILED_ENGLISH_COL.hasNext()
          if MATH == FAILED_ENGLISH_COL.getNext() then
               output(MATH)
          endif
     endloop
endloop
```

Topic 5 Pseudocode

Iterate through a stack

```
// Assume we have a stack S
// There will no more elements left in the stack after this
loop while not S.isEmpty()
      X = S.pop()
      output(X)
endloop
```

Iterate through a queue

```
// Assume we have a queue Q
loop while not Q.isEmpty()
      X = Q.dequeue()
      output(X)
endloop
```

Transfer from stack to queue

```
// Assume we have a stack S and an empty queue Q
loop while not S.isEmpty()
      X = S.pop()
      Q.enqueue(X)
endloop
```

Transfer from queue to stack

```
// Assume we have a queue Q and an empty stack S
loop while not Q.isEmpty()
      X = Q.dequeue()
      S.push(X)
endloop
```

Transfer from array to stack

```
// Assume an empty stack S
GRADES = [67, 92, 89, 55]
loop I from 0 to GRADES.length - 1
      S.push(GRADES[I])
endloop
```

Transfer from array to queue

```
// Assume an empty queue Q
GRADES = [67, 92, 89, 55]
loop I from 0 to GRADES.length - 1
     Q.enqueue(GRADES[I])
endloop
```

Transfer from stack to array

```
// Assume an empty array GRADES and a stack S
GRADES ARRAY
COUNT = 0
loop while not S.isEmpty()
     GRADES[COUNT] = S.pop()
     COUNT = COUNT + 1
endwhile
```

Transfer from queue to array

```
// Assume an empty array GRADES and a queue Q
GRADES ARRAY
COUNT = 0
loop while not Q.isEmpty()
     GRADES[COUNT] = Q.dequeue()
     COUNT = COUNT + 1
endwhile
```

Calculate an average of values in a stack

```
// Assume a stack S with an indeterminate number of values
SUM = 0
COUNT = 0
loop while not S.isEmpty()
     SUM = SUM + S.pop()
     COUNT = COUNT + 1
endloop
AVG = SUM/COUNT
output(AVG)
```

Calculate an average of values in a queue

```
// Assume a queue Q with an indeterminate number of values
SUM = 0
COUNT = 0
loop while not S.isEmpty()
     SUM = SUM + S.pop()
     COUNT = COUNT + 1
endloop
AVG = SUM/COUNT
output(AVG)
```

Search a 2D Array

```
// 5 rows of 4 values each
// GRADES[# of rows][# of columns]
GRADES[5][4]
// Search for position in 2D array of value X
input X
loop I from 0 to GRADES.length - 1
     loop J from 0 to GRADES[I].length - 1
          if GRADES[I][J] = X then
               output("Found")
               output("Position", I, ",", J)
          endif
     endif
endloop
```

Find average of elements in a 2D array

```
// Assume unknown number of elements
TEMPS[6][5]
SUM = 0
NO_OF_ELEMENTS = 0
loop I from 0 to TEMPS.length - 1
     loop J from 0 to TEMPS[I].length - 1
          SUM = SUM + TEMPS[I][J]
     endloop
endloop
// No of elements is 6*5
AVG = SUM/30
```

Topic 1: System Fundamentals

- Changing Software Systems (Considerations
  - Extent of change
  - Limitations of new system
  - Context in which the system will be used
    - Organizational Issues
    - Change in User Roles
- Changing Software Systems (Challenges)
  - Users don't like change
  - Some features may be omitted
  - Old systems may be faster
  - Incompatibility with other systems
  - Data loss
  - Expensive
- Changing Software Systems (Methods)
  - **Direct Changeover** - Old system stopped, then new started
    - Pros: Changeover swift, new system available immediately
    - Cons: No backup in case of failure
  - **Parallel** - Old and new system run concurrently - new data entered into both
    - Pros: Backup if new system fail, output from both system can be compared to verify that new system works correctly
    - Cons: Running both systems is expensive
  - **Pilot** - New system tested with small part of organization, bugs fixed, then expanded
    - Pros: All features tested before adoption by whole organization, staff who are part of pilot can train others, if failure, only small part suffers
    - Cons: No backup for pilot group in case of failure
  - **Phased** - introduced in phases, old system gradually phased out
    - Pros: Allows people to get used to new system, training can be done in stages
    - Cons: If system fails, no backup for that part of system
- Data Migration (Problems)
  - Moving data from one system to another
  - Can be huge process depending on sizes of systems
  - Necessary when transitioning to new system
  - **Possible problems**
    - **Incompatible file formats**
      - Different systems might use different versions of same software → different file formats
      - Data simply stored in different file formats because of different

software

        - **Data structure differences**
            - Ex: Data stored in arrays in one system and linked lists in the other
            - Ex: Data stored in tables in one system and spreadsheets in the other
            - Differences in how data is structured and stored
        - **Validations rules**
            - Different rules for what constitutes valid data
            - One system may be much less strict than the other, making data unacceptable for migration
        - **Incomplete data transfers**
            - The data transfer process is interrupted for some reason
            - Some data is moved to new system, but not all
            - Data may be lost
        - **Different data, currency, or character conventions**
            - Dates in different order in USA
            - Different currencies used in different systems
            - Different languages mean different characters used in different systems
- Legacy Systems
    - Old technology, computer system or application
    - No longer supported/available for purchase
    - Modernization may be expensive or time-consuming
    - Ex: Floppy disks, Windows XP
- Local vs. Remote (SAAS) Software
    - Local - runs on your computer (Notepad, Adobe Photoshop, etc.)
        - One-time fee
        - Installation
        - Update may not be automatic
        - Users may be using different version depending on update status
        - Can be used on one computer
    - Remote - Accessed through a web browser (Office 365, Google Docs, etc.)
        - Subscription fee
        - Can use used on any number of computers via web browser
        - Automatic Updates
        - Users always using the same version
- SAAS (Software-as-a-Service) Pros and Cons
    - Pros
        - Employees can operate software from anywhere

        - Cheaper
        - Fewer support staff necessary
        - Pay as you go/Subscription model
        - Scaling up to serve more users easy
        - Easier to maintain - don't need to worry about individual computer problems
    - Cons
        - Users have no control over availability of system
        - Users have no control over security
        - If SAAS provider gets acquired, user has no control over system
        - Large-scale data corruption possible
- Testing
    - **Static Testing**
        - Did we build the right system?
        - Examines documentation, planned specifications, test plans etc.
        - Makes sure that such documents will lead to the correct system being built
        - Verification of documentation
    - **Dynamic Testing**
        - Code is executed
        - Makes sure code produced required result
        - Checks for bugs
        - Software meets business requirements
        - **Validation** - makes sure software operates as expected
    - **Alpha Testing**
        - Testing done by internal team
        - Lasts for months
        - Very structures
        - Confirms software works as intended
        - Looks for bugs or UI issues to fix
    - **Beta Testing**
        - Used by potential end-users outside of company
        - Lasts for weeks
        - Largely unstructured, seeks to model real-world environment
        - Seeks to get feedback from users
        - Uses feedback to fix bigs and make software more useful for end-users
    - **Black Box Testing**
        - Tests software through using it
        - No access to codebase or internal workings
        - No programming knowledge required
    - **White Box Testing**
        - Tests a software program's code to make sure everything is working

        - Programming knowledge required; done by programmers
    - **User Acceptance Testing**
        - "End-user testing"
        - Product is tested by intended audience
    - **Automated Testing**
        - Uses automated testing software
        - Tests preconstructed (usually specified in software or written in code)
        - Program output automatically compared to expected output
        - Can rapidly complete complex/tedious tasks
- User Documentation (Types)
    - **Manual** (Paper manual, booklet, or pamphlet)
        - Pros: Doesn't require installation, computer, or internet connection
        - Cons: Can be damaged or lost, cannot be updated
    - **Online** (PDFs, website, video)
        - Pros: Can be much longer than manual, can be updated, search capabilities, can be updated easily
        - Cons: Internet required, can be difficult for inexperienced users to access
    - **Help files** (Locally accessible programs that display text)
        - Pros: Easily accessible in software program, contain general instructions for use, easier to access for inexperienced users
        - Cons: Requires installation first, which can be difficult for inexperienced users, lacks search capability
- User Training (Methods)
    - **In-person classes**
        - Pros: Cheap (many students for one instructor)
        - Cons: Less personalized, students may become bored, lost, or lose pace
    - **Online training** - Instructor training one student or small group of students
        - Pros: Can be more personalized, focuses on needs of students via online instructor
        - Cons: Can be more expensive if individual
    - **Self-instruction**
        - Pros: Low-cost (no teacher needed), flexible timeline, users can choose what to focus on
        - Cons: No guidance, users may feel lost, lack of structure means that users may not learn everything then need to
- Data Loss (Causes)
    - Hardware/System Malfunction
    - Human Error
    - Software Corruption
    - Malicious Software (Viruses, etc.)
    - National Disasters (Power cuts = equipment damage)
- Data Loss (Ways of Prevention)

        - **Failover Systems** - there is a secondary system that can be switched to if the primary system fails
        - **Redundancy** - a duplicate of a system's components and data are duplicated so that a backup is present
        - **Removable Media** - removal storage device can be used for data backup
        - Offsite/online storage - data is backed up at a different location or in the "cloud
        - **Physical Security** - data is secured from natural disasters or physical tampering
- Stakeholders
    - a person with an interest or concern in something (especially a business) (Oxford)
    - Ex: company owner, users, managers, shareholders, employees
- Stakeholders (How to Get Requirements for Software Functionality)
    - requirements - the tasks a program should be able to achieve (in the context of software)
    - Methods
        - **Direct Observation** - observe the system in use in the real-world
            - Pros: Opportunity to see users using a produce
            - Cons: Users may act differently under observation
        - **Interviews**
            - Pros: Asking users directly for feedback can be very insightful
            - Cons: Costly in times and resources
        - **Surveys** - Send users to questionnaires
            - Pros: Cheap and time-efficient
            - Cons: Hard to think of good question, Users may not take it seriously
- Prototypes
    - A plan or abstract representation of the end-product
    - Usually not fully finished
    - Displays functionality of one or two key aspects
    - Used to get feedback from stakeholders
- Iteration
    - Cyclical process that ends in a finished product
    - Process of prototyping, testing, analyzing test results, and refining product until requirements met
    - Allows constant improvement based on user or client feedback
- Software Deployment
    - **Release** - process of launching a new product
    - **Update** - software file that fixes a problem found after release
        - Bug
        - Security Vulnerability
    - **Patch** - Temporary fix between full releases
        - Bug
        - Security

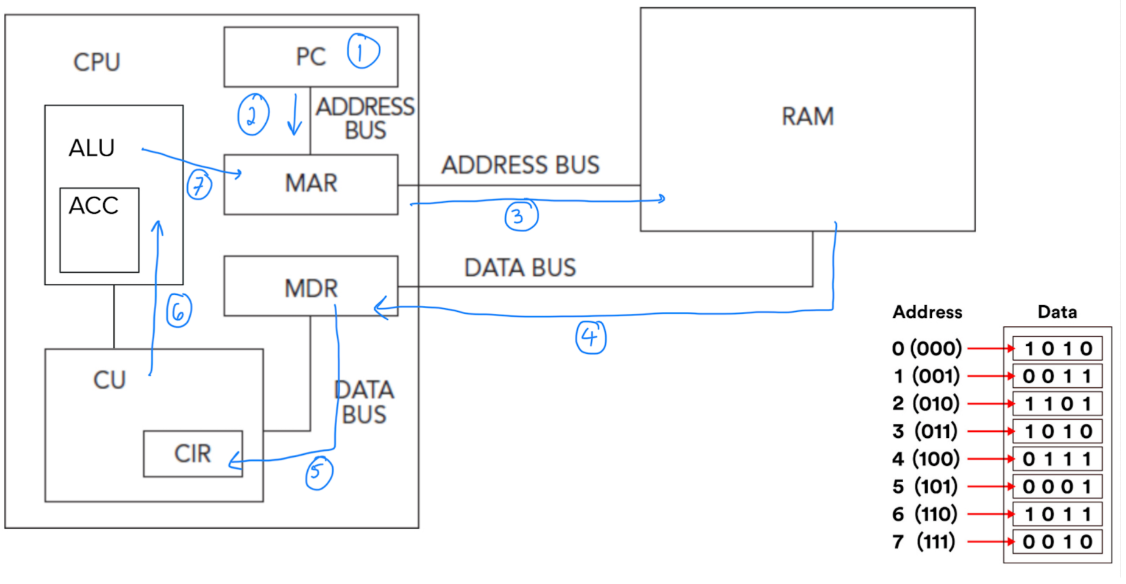
        - Upgrade capability
        - Upgrade Driver
- Usability - How effectively and efficiently a product can be used
- Usability Problems (Examples)
    - Laptop - battery life too short
    - Phone - screen size too small, not waterproof
    - Digital camera - buttons too small, software buggy
- Accessibility - How easily people can use software
- Accessibility (Examples of how to Improve)
    - Braille Keyboard
    - Touch Screen
    - Voice Recognition
    - Text Dictation

Topic 2: Computer Organization

- CPU (Central Processing Unit)
    - “The Brain”
    - Process all instructions as binary code
    - Code is executed here
    - Arithmetic, logical, input output operations
    - Directly connected to RAM (Random Access Memory)
- Fetch-Decode-Execute Cycle (**aka** Machine-Instruction Cycle, Von Neumann Model)
    1) Address to be checked comes up in PC.
    2) PC sends the next address to be checked to the MAR.
    3) MAR sends address to RAM
    4) Data at address is sent to MDR.
    5) MDR sends data to the CIR (Current Instruction Register) for decoding.
    6) CIR passes the decoded instructions on to the ALU, which does everything that needs to be done.
    7) ALU gives address to MAR for any further instructions.
    - Processor speed = # of FDE Cycles per Second (2GHz = 2 billion cycles per second)
    - FDE Cycle by by CU
    - # of processor cores = # of ALUs

FDE = Fetch-Decode-Execute
PC = Program Counter
MAR = Memory Address Register
RAM = Random Access Memory
MDR = Memory Data Register
CU = Control Unit
ALU = Arithmetic Logic Unit

CPU (Von Neumann Model)



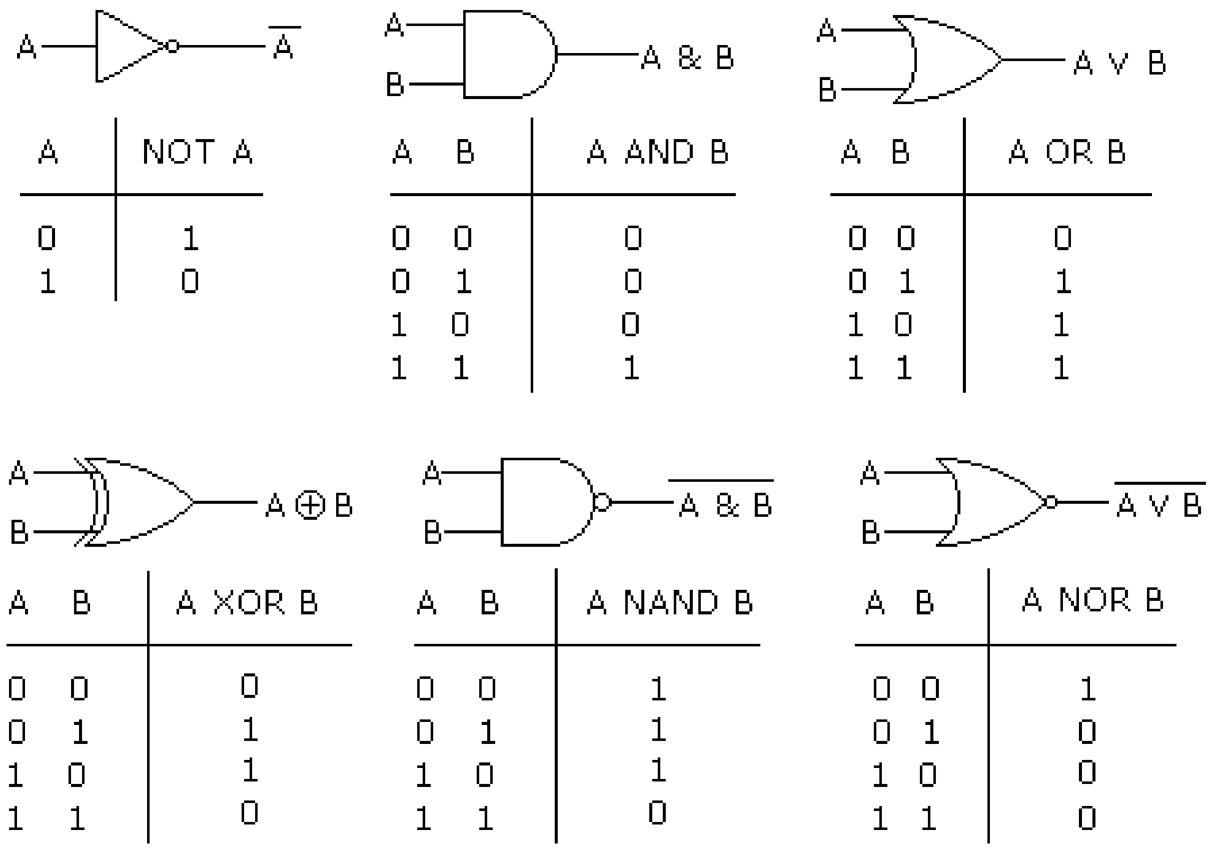
Address	Data
0 (000)	1 0 1 0
1 (001)	0 0 1 1
2 (010)	1 1 0 1
3 (011)	1 0 1 0
4 (100)	0 1 1 1
5 (101)	0 0 0 1
6 (110)	1 0 1 1
7 (111)	0 0 1 0

- Primary Memory
  - RAM (Random Access Memory)
    - “Short-term memory”
    - Consists of units of data, each with a hexadecimal address
    - Volatile - data lost when electricity is cut off (when computer is turned off) (non-persistent storage)
  - ROM (Read-only Memory)
    - Used to store permanent instructions necessary to boot computer
    - Holds BIOS (Basic Input Output System)
    - Instructions written in factory
    - Non-volatile (data persists without electricity)
- Cache
  - Stores frequently used instructions from RAM
  - Processor checks cache first
  - Increases speed of FDE cycle
  - Cache memory more expensive, but faster
  - Closer to CPU than RAM, which contributes to speed
  - L1 (Fastest) and L2 Cache
- Secondary Memory
  - Hard Drive, Removable Storage (USB Drive, CDs)
  - “Long-term Memory”
  - Persistent storage
  - Holds all data not currently in use
  - Data for running programs transferred from secondary to primary memory

        - Slower and cheaper than primary
        - Not directly connected to CPU
        - Much larger amounts
- Virtual Memory
    - When primary memory overloaded, data sent to secondary
    - Slower
    - Temporary
    - Returned to primary memory as needed
    - Stored in units called “pages”
- Operating Systems
    - “set of software that controls computer’s hardware and resources and provides services for computer programs”
    - 5 roles
        - **User interface**
        - **Memory Management**
        - **Peripheral Management**
        - **Multitasking**
        - **Security**
    - User Interface
        - Link between user and hardware
        - 4 types of UI
            - **GUI (Graphical User Interface)** - menus, point and click
            - **CLI (Command Line Interface)** - type in commands
            - **NLI (Natural Language Interface)** - speak to computer (Siri)
            - **MBI (Menu Based Interface)** - Like CLI, but no commands, only menu option
    - Memory Management
        - Keep track of storage devices (HDD, Flash Drive)
        - Allocate memory (RAM) to programs
        - Modify memory locations
        - Sort data on disk drives and RAM for efficiency
        - Organize data into folders
        - Copy and delete files
    - Peripheral Management
        - Peripherals - keyboard, mouse, monitor, etc.
        - Coordinate with BIOS (basic input/output system)
        - Use device drivers to interface with peripherals
        - Device drives translate peripheral signal
    - Multitasking
        - Allocates CPU cycles to concurrent programs based on priority and time
        - Each program given a slice of time, or a “turn” to use CPU
        - Slices vary in length of time

- Security
    - Username and password
    - User permissions
    - File permissions for reading and writing
- Application Software
  - Word Processors
  - Spreadsheets
  - Database Management System (DBMS) (MS Access)
  - Email Client (Outlook)
  - Web Browser
  - Computer Aided Design (CAD)
  - Graphic Processing Software (Photoshop)
- Units of Data Storage
  - 1 byte = 8 bits
  - 1 kilobyte = 1024 bytes
  - 1 megabyte = 1024 kilobytes
  - 1 gigabyte = 1024 megabytes
  - 1 terabyte = 1024 gigabytes
- Binary
  - Language of CPU and computing
  - 1, 0 are the only possible digits
  - Base 2 - Each digit corresponds to a value with a base of 2
- Hexadecimal
  - More efficiently represents large binary values
  - Uses
    - Displaying Colors
    - Memory addresses (esp. in assembly)
    - MAC Addresses
  - Base 16
- Conversions to Know
  - Binary → Decimal
  - Decimal → Binary
  - Hexadecimal → Binary
  - Binary → Hexadecimal
  - Hexadecimal → Decimal
  - Decimal → Hexadecimal
- Displaying Text
  - Strings are made up of characters
  - Each character is represented by a certain number of bytes
  - Two formats: ASCII and Unicode
  - ASCII
    - Use 8 bits for each character

        - 7 bits used for each character, 1 bit is parity bit
        - 2^7 or 128 possible characters
        - Lower case, capitals, numbers, symbols, spaces, punctuation included
    - Unicode
        - Used to represent multiple languages
        - More bits, so more characters can be represented
        - UTF-8 → 8 bits, UTF-16 → 16 bits, UTF-32→32 bits
- Displaying Images
    - Each display (monitor, phone screen, etc.) is divided into pixels
    - pixel = "little square" in grid
    - pixel = smallest controllable element in display
    - Each pixel has a different color
    - Each pixel has one color at one moment
    - The color is combination of red, blue and green
    - Pixels together display image
    - 1024x764 screen resolution →1024 pixels high, 764 wide
    - 1080p, HD, 4K refers to number of pixels in screen
- Pixels + Colors
    - Each color in each pixel represented by 6 digit hexadecimal value(Ex: 70EF5A)
    - First two values represent red, second two represent green and last two represent blue (Ex:70EF5A)
    - 16^2*16^2*16^2 = 256*256*256 possible color combinations
- Logic Gates
    - Processors/chips made up of millions of "switches" called logic gates
    - Each logic gate takes 1-2 inputs and 1 output
    - Possible inputs are 1 or 0
    - 1 = high voltage, 0 = low voltage
    - Voltage = "pressure that forces charged electrons" to flow through circuit



A	NOT A
0	1
1	0

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

Logic Gates Present on IB Exam

Topic 3: Networks

- Server - a computer system or a software application that provides a service to other computer systems on the same network
- Client - computer system or software application that requests a services from a server connected to the same network
- Computer Networks
    - A computer network is formed when 2 or more computers are linked together.
    - Every computer or device on a network can send and receive data from any of the other computers or devices connected to the network.
- Devices to Connect Computers
    - **Hub**
        - Connection point for devices on a single network
        - When a network device wishes to send data to another device on the network, it copies the data and sends it to all devices connected to its ports.
        - This generates a lot of unnecessary traffic on the network, slowing it down
    - **Switch**
        - Similar to a hub, but can identify which device is connected to which port, allowing a network connected by switches to operate much faster
        - data is only sent to the computer that needs it
    - **Router**
        - Can connect multiple networks and serves as an intermediary between them (i.e. home network and Internet)
- Internet
    - Globally connected network system
    - Uses TCP/IP protocol to transmit data
    - No centralized governance
    - Hosts the web pages that make up the World Wide Web
    - ISPs (Internet Service Providers) run networks that provide internet access
    - ISPs are "glued together" by internet exchanges (IXPs)
    - IXPs are the key to connecting all the different ISPs and networks together - usually run and maintained by nonprofits
- How the Internet Works
    - Data travels in the form of **packets**.
    - Every file you send or request you make is split up into packets over the internet.
    - Individual packets often take different routes through exchanges, ISPs, and junction boxes
    - Ultimately reach the same destination and are reassembled at destination.
- Packet Switching
    - A packet is a unit of information suitable for travel through computer networks.
    - Data is grouped into packets.

        - A file being transmitted through packet switching may be divided into multiple packets and each packet could follow a different route to the same destination.
        - This allows for more efficient data transmission and reduced delays.
- Router
    - Used to manage traffic
    - Controls the flow of data packets
    - Checks address of data packets
    - Puts packets on correct path
    - Secure transmission
- Types of Networks
    - LAN
        - Covers a single building or collection of buildings
        - Less than 1KM radius
    - WAN
        - Covers more distance than LAN
        - Greater than 1KM radius
        - Often uses multiple routers
        - May be several LANs connected together
        - Examples: Internet, Cellular Network, ATM network
    - VLAN
        - 1 or more physical LAN (local area network)
        - Network switch used to divide network
        - Works as multiple networks
        - More than the number of physical networks
        - Each VLAN may have different security requirements
        - Appears to an outsider as 1 LAN or WAN
    - SAN
        - Network of storage devices
        - Often appear as attached drives
        - Purpose specific servers (email server, application server, database server, storage server, etc.)
        - Backup servers, battery backup in case of disasters
        - Not accessible through LAN
        - Better performance
    - WLAN
        - LAN by which devices are connected by high-frequency data waves
        - Requests and responses from the internet (internet access) can be made through WLAN
        - Uses can move around
        - “Wifi network”
        - Uses type of radio waves called Wifi (Wireless Fidelity)
    - Intranet

    - Private network
    - Sort of a “private internet”
    - Uses TCP/IP
    - Can view web pages similar to those on the internet
    - Intranet web pages only accessible from computers on network
    - North Korea - nationwide intranet
- Extranet
    - If the intranet system is accessible through an online portal, it becomes an **extranet**.
- VPN
    - You use a VPN client, like NordVPN or something similar.
    - From your VPN client, you send all your requests, like www.google.com to a VPN server
    - The VPN server makes a request to that internet resource on your behalf and the VPN server returns the response to you via the VPN client
    - Connection with VPN server encrypted.
    - The process of creating and maintaining an encrypted connection between the server and client is called **tunneling**.
    - This encryption is done using an **encryption** technology like SSL, TLS, or IPSec.
- VPN (Advantages)
    - **Authentication**
        - Encryption cannot be changed by outsider
    - **Encryption**
        - Data cannot be accessed or altered
    - **Tunneling**
        - Data can be be secured using proprietary protocols of company or external VPN provider
    - **Multiple Exit Nodes**
        - Origin of initial data packets unclear
        - Data can appear to originate from multiple IP Address

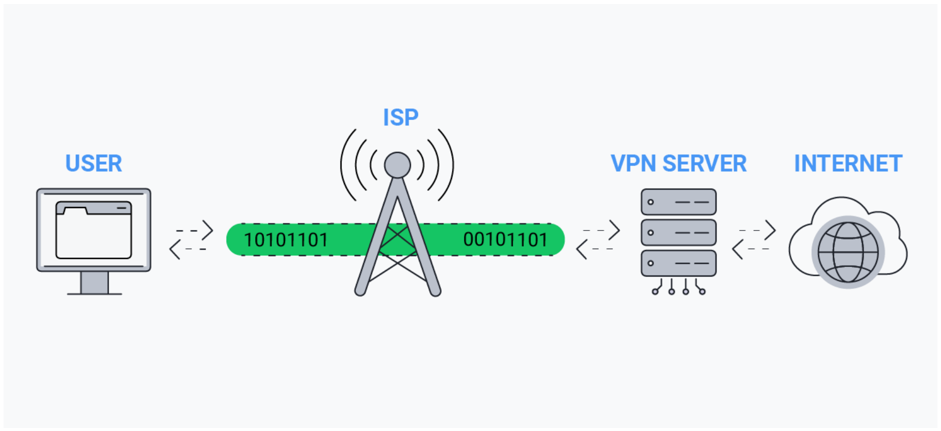


Illustration of VPN Functionality

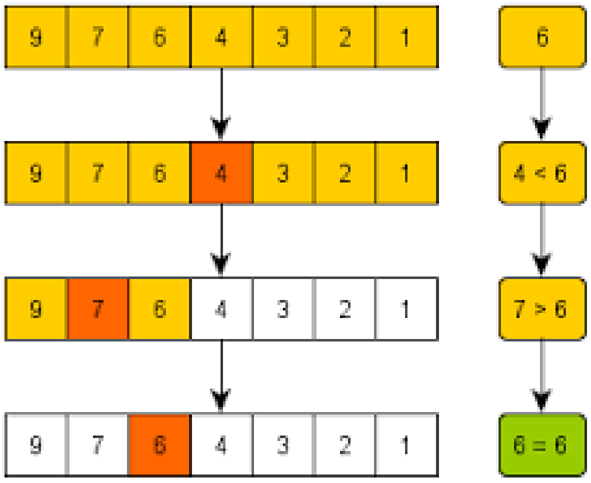
- PAN
- P2P
    - Allows 2 or more computers to share resources with each other
    - Each computer acts as both a client and server
    - No central point
- Protocols
    - A protocol is a set of rules for data transmission (data packets)
    - Examples
        - TCP/IP - Rules for transmitting data on the internet
        - HTTPS - Rules for securely transmitting data between a web browser and a server
        - SFTP - Rules for securely transferring files from client to server
- Protocols (Roles)
    - **Maintain Data Integrity** - same data that is sent is received
    - **Flow Control** - data is sent and received at the same rate
    - **Prevent Deadlock** - two packets don't "block each other"
    - **Prevent Errors** - Make sure no errors introduced in transit
- OSI (Open Systems Interconnection) Model
    - Models network connection
    - Layers (Memorize 3) - Application, Presentation, Session, Transport, Network, Data Link, Physical

- Transmission Media
    - Physical media by which data packets are transmitted
    - Two types - wired and wireless
    - Wired - Ethernet, Fiber Optic, Coaxial Cable
    - Wireless - Radio Waves (WLAN), Microwave, Wireless
    - 4 Factors to Consider
        - **Security**
        - **Reliability**
        - **Cost**
        - **Speed**
- Wifi vs Ethernet
    - Ethernet faster, more consistent speed, lower latency data transfer
    - Wifi easier to install and deploy, cheaper
    - Ethernet more secure, can only be tapped through physical access
- Fiber Optic vs. Copper Cabling
    - Fiber - faster, more expensive, lighter, almost impossible to tap, immune to interference, longer range(40 km+)
    - Copper - Susceptible to electromagnetic interference, shorter range (100m), heavier, thicker
- Transmission Speed
    - Primary Factors
        - Traffic
    - Secondary Factors
        - Time of Day
        - Distance
        - Infrastructure
    - Tertiary Factory
        - Environmental Issues (Temperature, Interference)
        - Financial Factors (Cheaper equipment, etc.)
        - Type of Data (Size, streaming, etc.)
- Compression
    - **Lossy Compression**
        - Removes data
        - Smaller file size
        - Irreversible
        - Used when some data can be lost (videos, images, etc.)
    - **Lossless Compression**
        - Uses algorithm to reduce file size
        - Larger than lossy, but smaller than normal
        - Reversible
        - Used when no data can be lost (text, software, etc.)

- Authentication
    - One Factor
        - One way (something you know)
        - Examples - password, security question, or PIN code
    - Two Factor
        - Two ways (something you know, something you have)
        - Examples - smartphone, smart card, hardware token, in addition to first factor
    - Three factor
        - Three ways (something you know, something you have, something you are
        - Examples - fingerprint, retinal scan, facial recognition, in addition to first two factors
- Encryption
    - Encodes data
    - Data can only be read by sender and receiver
    - Involves a “key”
- MAC (Media Access Control) Address
    - Used to identify network-enabled devices in networks
    - “Hard-coded” by manufacturer
    - Network interface controller
    - Unique, 6 pairs of 2 hexadecimal digits
    - Check against white list
- Firewalls
    - Can be hardware- or software-based
    - Analyzes data packets
    - Controls incoming and outgoing traffic according to predetermined rules
    - Can filter based on MAC addresses
- Physical Security
    - Locked doors
    - Cages
    - Security Guards
    - Secure Rooms
    - Hurricane/Earthquake Proof
    - EMP-Insulated

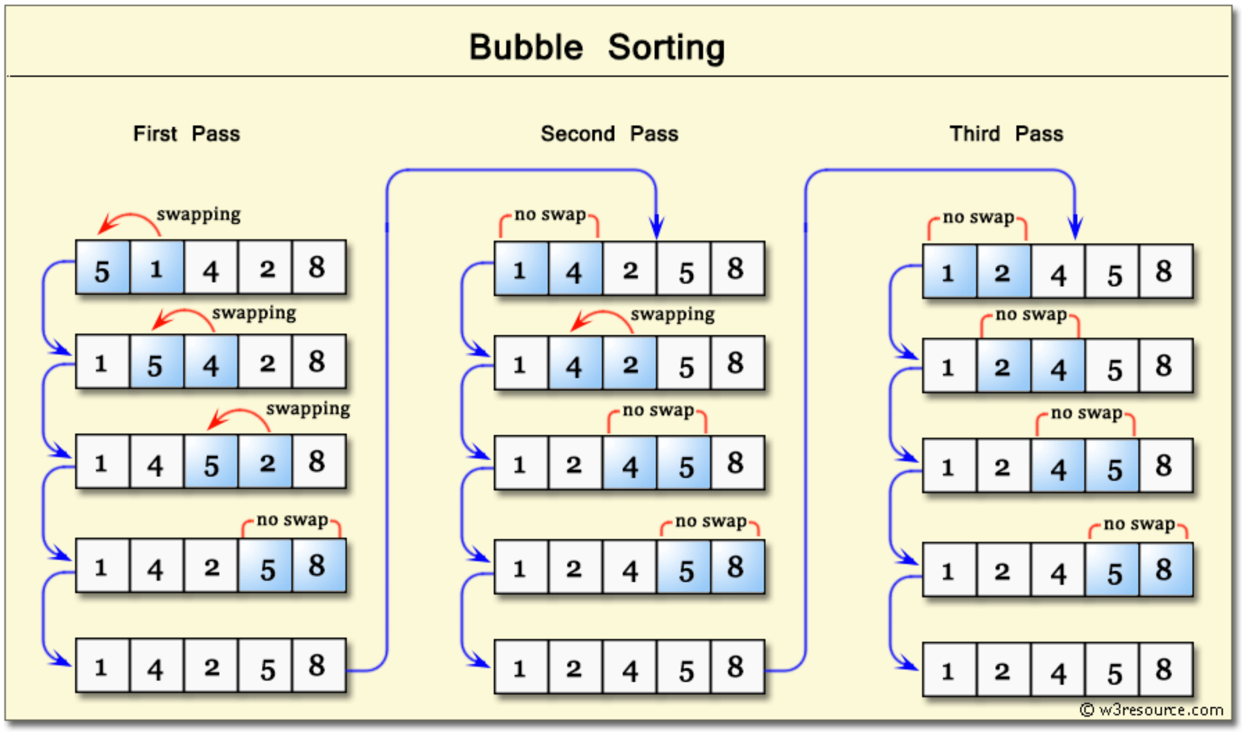
Topic 4: Programming

- Linear Search (Sequential Search)
    - Progress from beginning to end of array and check whether each element is equal to the target value
- Binary Search
    - Can only be used with an array that is already sorted (by ascending or descending order)
    - Process (assuming an array is used)
        1) Compare the target value with the value of the middle element of the array.
        2) If the values match, the value being searched for was found.
        3) If the target value is less than the middle elements of the array, then the action repeats for the value (sub-array) to the left of the middle element.
        4) If the target value is greater than the middle element, it repeats on the right of the middle element.
        5) If the remaining array to be searched is empty, the target value was not found.

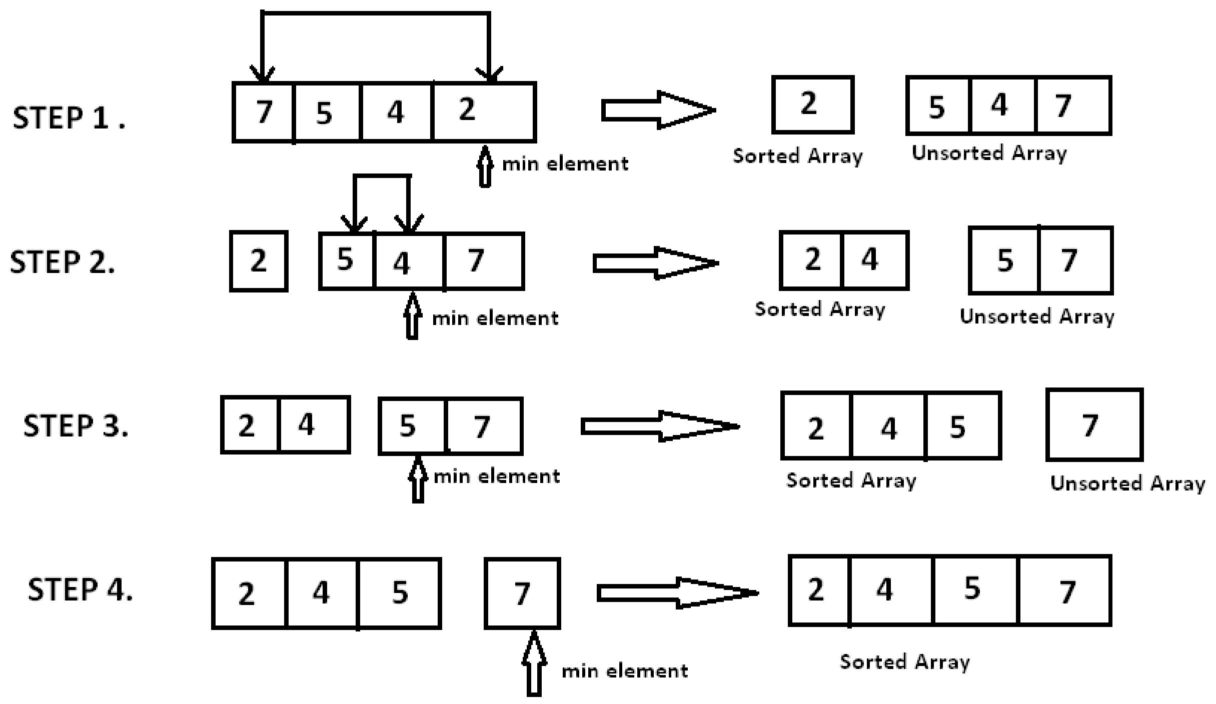


Binary search with target value of 6

- Bubble Sort
  - Goes through the array and compares each pair of elements.
  - If ascending, swaps elements where the first element is greater than the second. Otherwise, if descending, swaps elements where the first is less than the second.
  - Repeats the process, iterating through the entire array as many times as is necessary until the array is sorted.
  - Array will be traversed a maximum of n - 1 times, where n is the length of the array



- Selection Sort
    - Compared to bubble sort, inefficient with large numbers of elements
    - Process
        - Divides array in two parts - subarray of elements already sorted and subarray of those yet to be sorted
        - Sorted sublist is on the left and is initially empty
        - Algorithm goes through the unsorted portion, finds the minimum, and swaps it with the leftmost unsorted element
        - This process repeats until the sorted portion is empty.



Topic 5: Abstract Data Structures

- Types of Data Structure
    - Static
        - Number of elements determined at creation of data structure - cannot be changed later on
        - Can be memory inefficient, because RAM is allocated based on predetermined number of elements, rather than actual elements, leaving possibility of unused memory space
        - Iterated through using FOR loops
        - Examples: Arrays
    - Dynamic
        - No need to predetermine number of elements, any number of elements can exist in data structure at any time
        - Each element exists at a particular memory address and is referred to as a **node**.
        - Each node contains a "reference" to the memory address of the next element in the data structure. This reference is called a **pointer**.
        - Usually more memory efficient than static data structures
        - Iterated through using WHILE loops
        - Examples: Collections, Stacks, Linked Lists

Static	**Dynamic**
Predetermined number of elements	Any number of elements possible
Stored in one contiguous block of memory	Individual nodes can be stored separated all over memory
If empty positions in array, memory inefficient	Memory efficient
Iterate using FOR loop	Iterate using while loop

- Recursion
    - Recursion is when a method calls itself until some terminating condition is met, known as the **base case**.
    - Recursion can be used in situations where the next result of an operation depends on the current result.
    - Some good examples including the Fibonacci sequence or a factorial function
    - Each step until the base case is reached is called a **recursive step**.
    - All recursive functions can be written iteratively (without calling itself)

Advantages	Disadvantages
Code is cleaner and elegant	Recursive logic can be hard to follow
Complex task can be broken down into repetitive series of simpler subproblems	Usually more inefficient due to memory and time overhead of repeated function calls
Sequence generation is easier than with nested loops	Harder to debug

- Handling Recursion Problems
    - Start with initial parameter(s)
    - Write down all the return statements, until you get to the point where the parameter matches your base case.
    - At this point, you can turn your function call into an actual number and work your way back upwards

- Recursion Example 1

```
def factorial(num):
    if num == 0:
        return 1;
    return num*factorial(num-1)
factorial(4)→
```

```
    return 4*factorial(3) = return 24
        return 3*factorial(2) = return 6
            return 2*factorial(1) = return 2
                return 1*factorial(0) = return 1
                factorial(0) = 1
```

So, factorial(1) = 1*1 = 1, factorial(2) = 2*1 = 2, factorial(3) = 3*2 = 6, **factorial(4) = 24**

- Recursion Example 2

```
mistier(n)
    if ( n == 0):
        return 1
    else:
        return 4 * mistier(n-1) + 2
mistier(5)→
```

return 4 * mistier(4) + 2 = return 1706

    return 4 * mistier(3) + 2 = return 426

        return 4 * mistier(2) + 2 = return 106

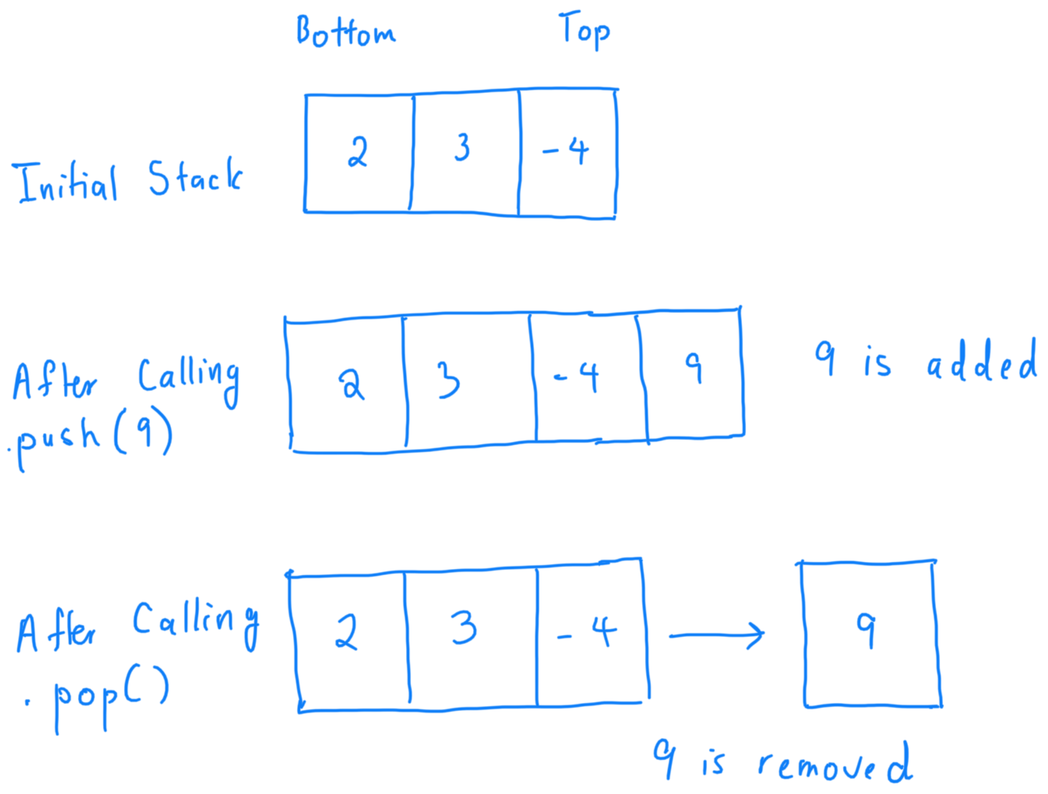
            return 4 * mistier(1) + 2 = return 26

                return 4 * mistier(0) + 2 = return 6

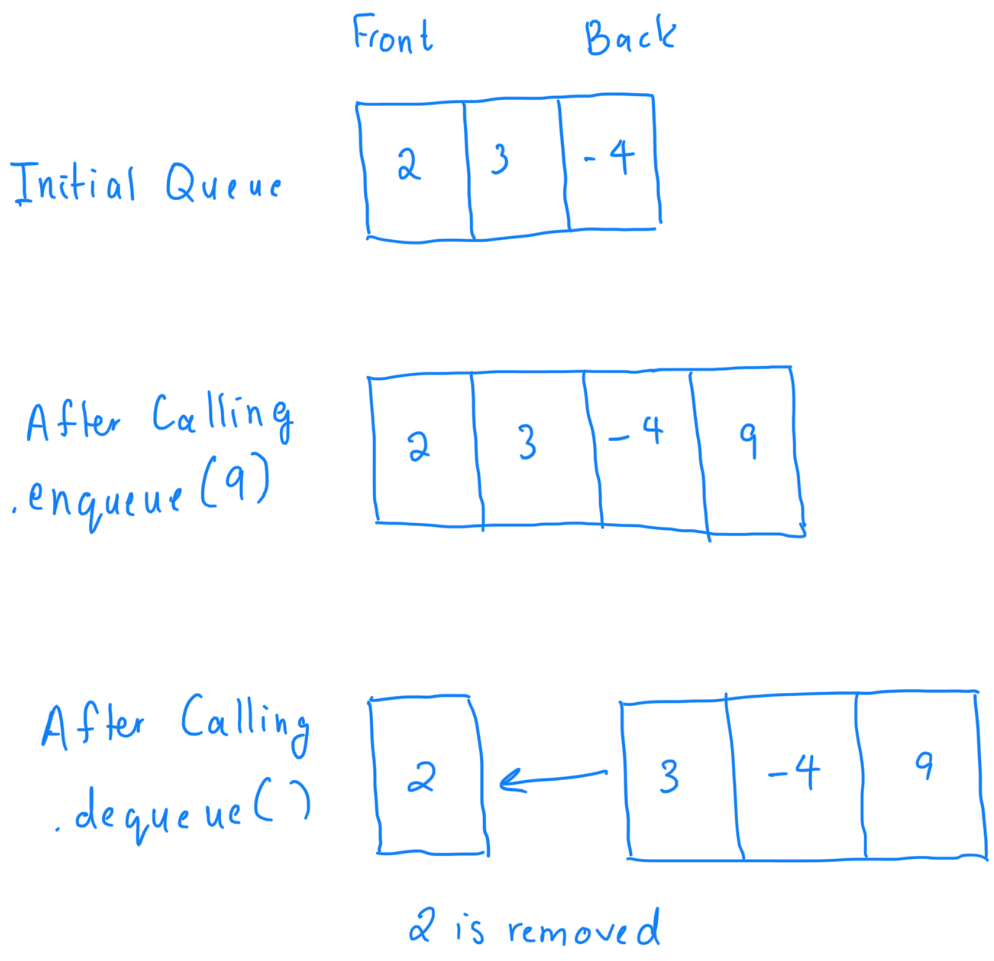
                    mistier(0) = 1

So, mistier(1) = 6, mistier(2) = 26, mistier(3) = 106, mistier(4) = 426, **mistier(5) = 1706**

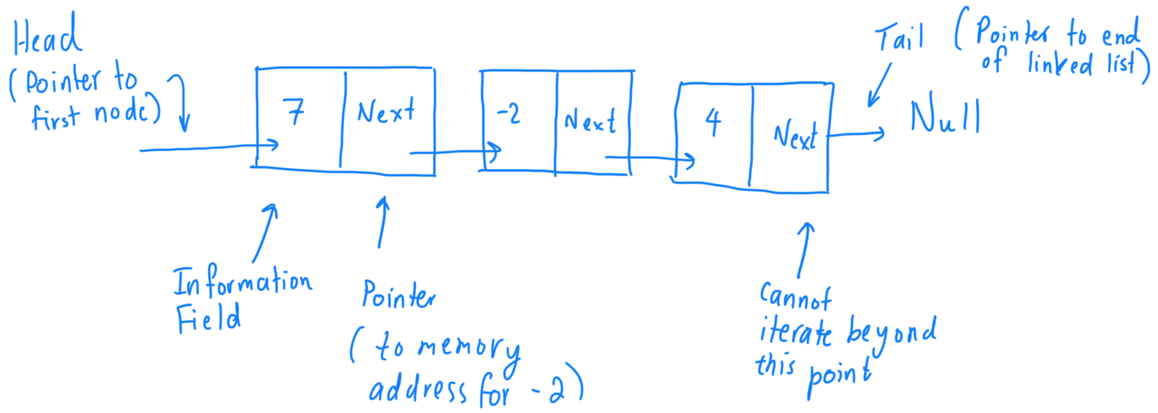
- Stacks
    - Dynamic Data Structure
    - LIFO (Last in First Out) - the last element to be "pushed" (inserted) into the stack is the first to be removed ("popped")
    - Used to keep track of function calls



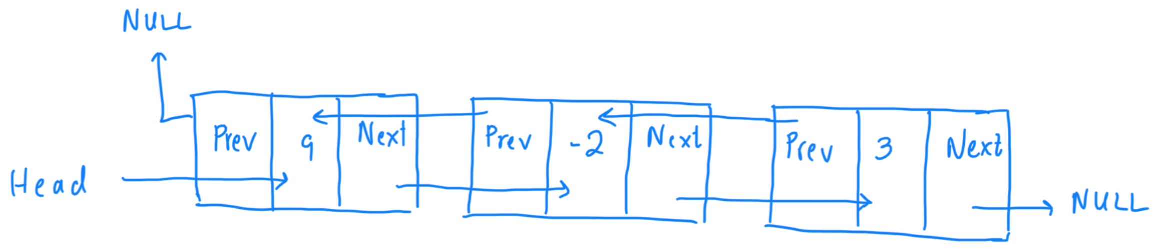
- Queues
    - Dynamic Data Structure
    - FIFO (First in First Out) - the first element to be “enqueued” (inserted) is the last to be “dequeued” (removed) like a real-life queue



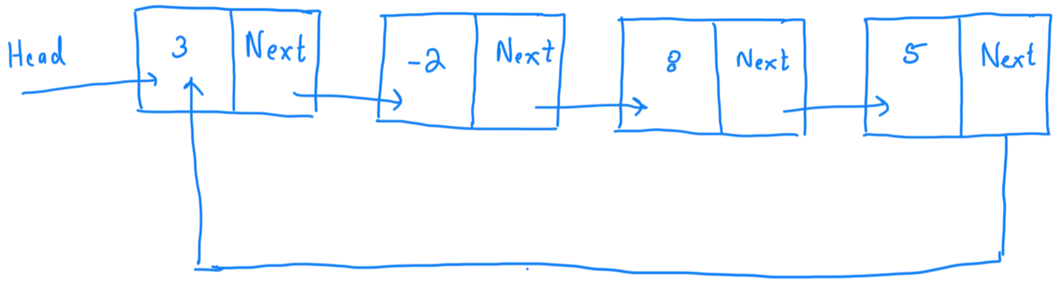
- Linked Lists
    - Singly Linked List
        - Accessed using the **head**, which is a pointer to the first element of the linked list
        - A **pointer** leads to a memory address in RAM
        - Every element has a **pointer** to the next element and an **information field**, which contains the actual data
        - Can only move one way through the linked list
        - Stop at the last element, cannot progress any further
        - The last pointer is called the tail, which leads nowhere


Head
(Pointer to first node)
7
Next
-2
Next
4
Next
Tail (Pointer to end of linked list)
Null
Information Field
Pointer
(to memory address for -2)
Cannot iterate beyond this point

    - Doubly Linked List
        - Accessed using the head, which is a pointer to the first element
        - Each element has pointers to both the next and previous element, so it's possible to move both forwards and backwards through the linked list
        - The first and last elements have one pointer that points to the next or previous element and the other leads nowhere

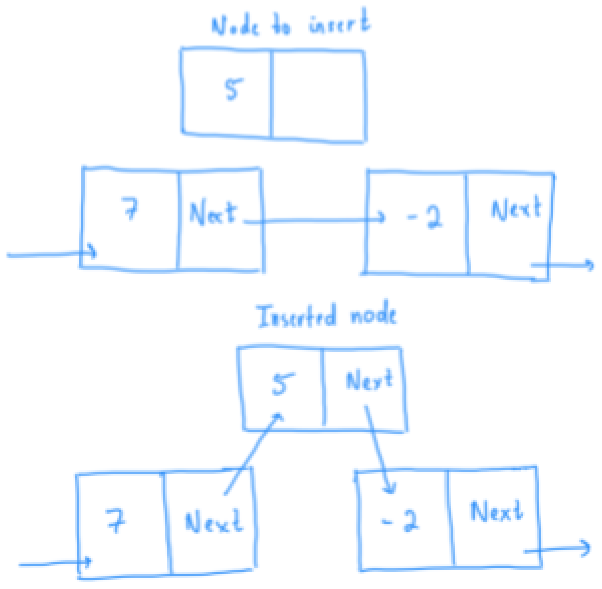


- Circular Linked List
    - Accessed using the head, which is a pointer to the first element
    - Can only move in one direction, but the last element has a pointer to the first element, so you constantly move in a "circle"

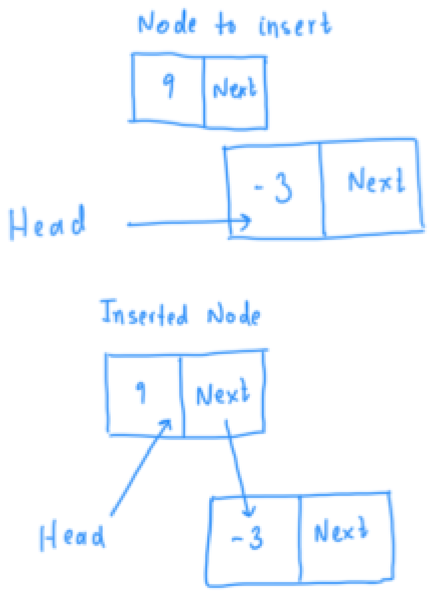


- Adding Elements to Linked List
    - Add elements by pointing existing pointers to new element and by pointing the new element to previous and/or subsequent elements depending on the type of linked list

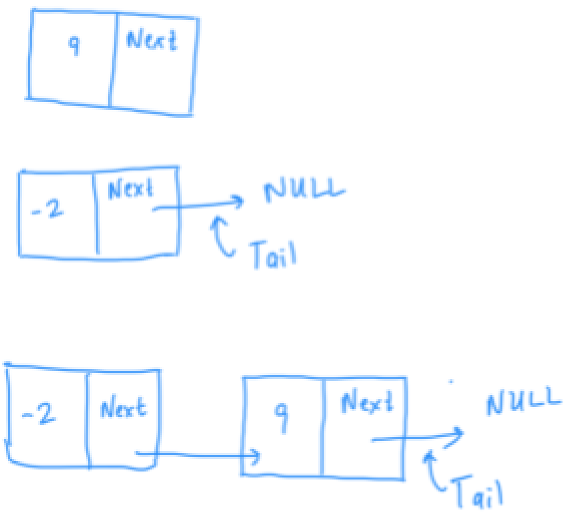
Insert node into middle of linked list



Adding new node to beginning of linked list

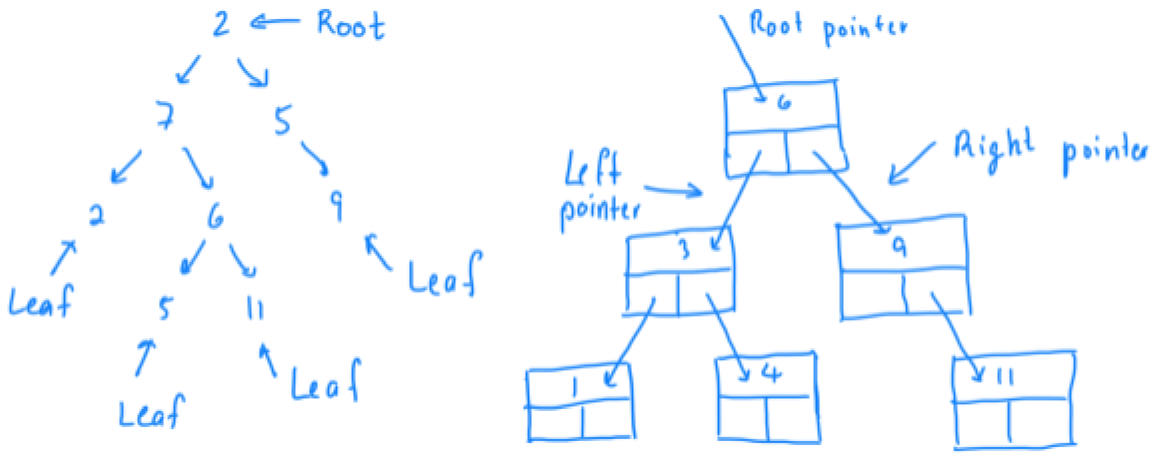


Adding new node to end of linked list



- Binary Trees (aka Binary Search Trees)
    - There is a pointer to the first node, which is at the top and called the **root node**
    - Each node points to two other **child nodes**, one of which has a higher value and one of which has a lower value than the parent node.
    - Nodes that point to no other nodes are **leaves**.
    - All nodes to the left of the parent node will have numerical values lower than the parent node and all nodes to the right will have higher values
    - You can generate a binary tree by following this rule, adding values to the tree in the order given.

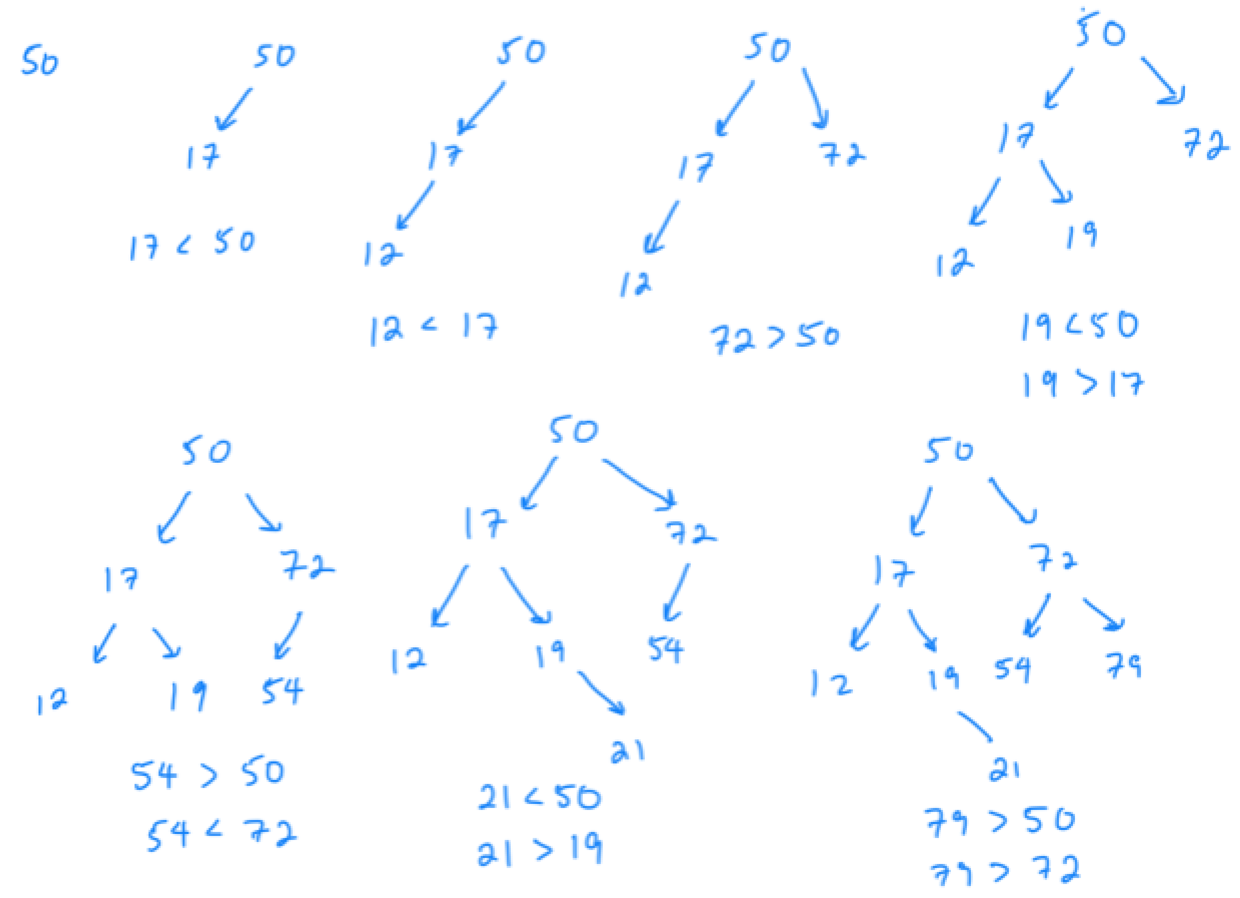
Anatomy of a Binary Tree



    - The structure of a binary tree mimics a binary search. You can find data in less than half the time required with a data structure like an array.

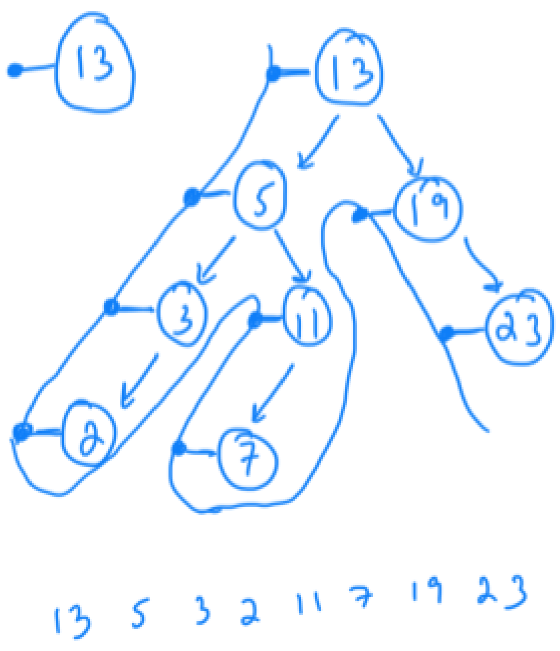
Generating a binary tree from the set of integers {50, 17, 12, 72, 19, 54, 21, 79}

50 17 12 72 19 54 21 79

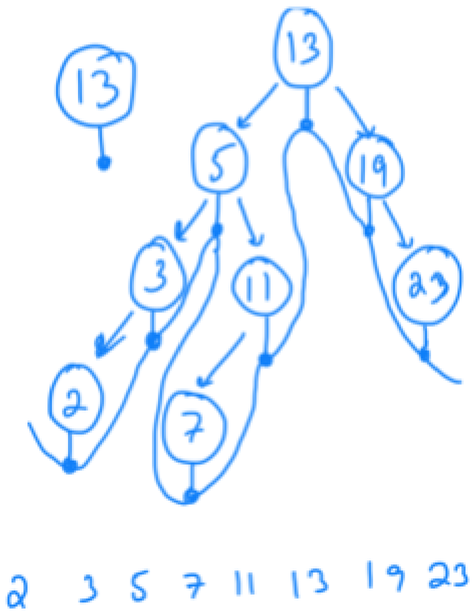


- Traversing Binary Trees
    - 3 Ways: Preorder, Inorder, Postorder
    - The order can be ascertained by drawing a flag or a line from each element to left (preorder), to the bottom (inorder), or the right (postorder)

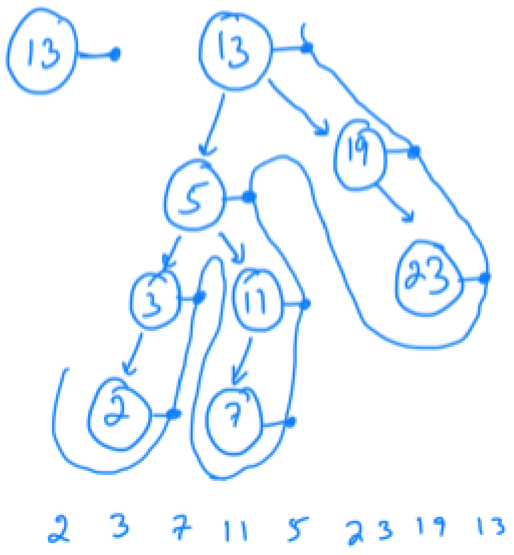
Preorder



Inorder

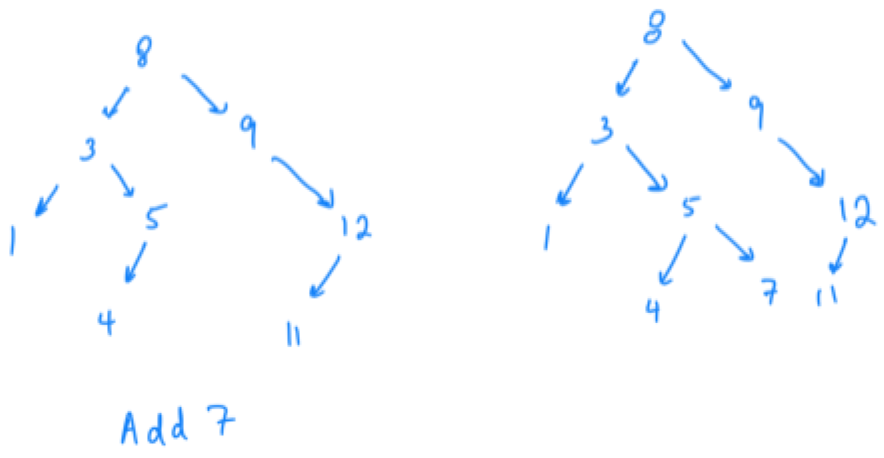


Postorder

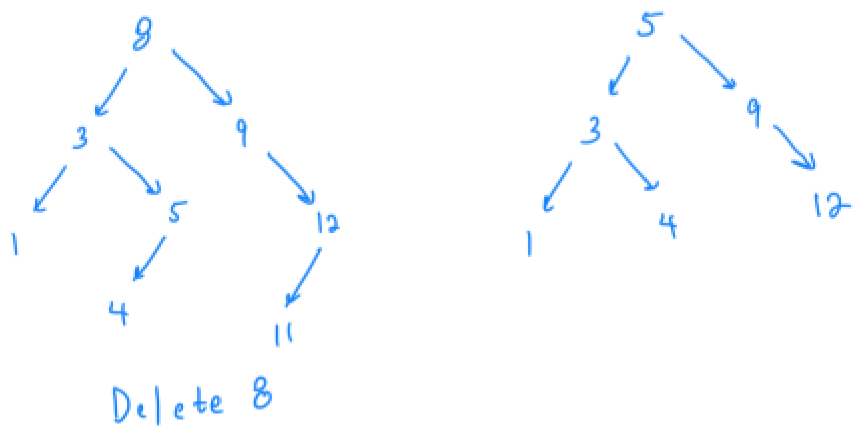


- Inserting and Deleting Values
    - When inserting values, you can simply pretend that your new number is the next number in the initial sequence and add it to the appropriate position
    - When deleting values, you must remove the value, and then reconfigure the binary tree where necessary so that you are abiding by the the order in which numbers were added by the binary tree, but so that each element has numbers less than it on the left and greater than it on the left.

Inserting a Value into a Binary Tree



Deleting a Value from a Binary Tree



Topic 6: Resource Management

- Resources to Manage
  - Primary Memory (RAM)
    - **Role** - Stores data for currently running programs
    - **Common Capacities** - 4, 8, 16 GB
    - **Effect of Limited Primary Memory**
      - Fewer programs or processes can run simultaneously
      - Computer must rely on virtual memory, which is much slower
  - Secondary Storage
    - Role
      - Stores all data, including the OS, program files, and multimedia data
      - Used as virtual memory when RAM is overloaded
    - Common Capacities
      - HDD (Hard Disk Drive) - 500GB, 1TB, 2TB
      - SSD (Solid State Drive) - 256GB, 512GB
      - Optical - CD: 650MB, DVD: 4.7GB
    - Effect of Limited Secondary Storage
      - Limited Virtual Memory from RAM to rely on
      - Limited number of programs and files can be stored and used
  - Processor (CPU)
    - **Role**
      - Handles all calculations and logical operations
      - Can contain multiple cores (ALUs) to conduct more operations per second
    - **Common Processor Speeds**
      - Speeds measured in GHz (1 billion FDE (Fetch-Decode-Execute) cycles per second)
      - Common Speeds: 2, 2.4, 3.2 GHz
    - **Effect of Limited Processor Speed**
      - All programs run more slowly
      - Any calculation-intensive operations (games, video editors, graphic design software, etc.) run more slowly
  - Bandwidth (Network Transmission Capacity)
    - **Bandwidth (Bitrate)** - how much data can be sent at the same time in a given time frame
    - **Common Speeds**
      - Speed expressed in megabits or gigabits per second
      - 100 Mbps = 100 megabits per second = 100/8 (12.5 megabytes per second)

        - Broadband (16-100 Mbps), LAN (1 Gbps)
    - **Effect of Limited Bandwidth**
        - Slow data transfer across networks
        - If accessing internet through network, internet speeds will be slower
- Screen Resolution
    - **Screen Resolution** - number of pixels that can shown horizontally and vertically (Usually defined by height x width of screen)
    - **Common Screen Resolutions**
        - 1024 x 768 (XGA)
        - 1366 x 768 (HD 720p)
        - 1920 x 1080 (HD 1080p)
        - 4096 x 2304 (4K)
    - **Effect of Limited Screen Resolution**
        - Images and videos are pixelated
        - Smaller file sizes required to display images and videos, but quality is much lower
- Sound Processor
    - **Role**
        - Translates analog audio signals (microphone) to digital signals and digital signals to analog audio signals (speakers)
        - Process audio files to produce high quality analog output
        - Frees up CPU from audio-related tasks
    - **Common Use Cases** - Computers, Home Theatre Systems, Phones
    - **Effect of Limited Sound Processing Capability**
        - Low quality audio output
        - CPU capacity is lower, because it is being utilized for audio-related tasks
        - Audio-related operations take longer
- Cache
    - **Role**
        - Sits between CPU and RAM
        - Contains instructions/data most frequently requests by CPU from RAM
        - Speeds up CPU operations (specifically FDE cycle)
    - **Common Cache Sizes/Types**
        - 8MB, 16MB, 32MN
        - Certain amount of cache attached to each core
    - **Effect of Limited Cache Size**
        - CPU must wait longer to receive instructions from RAM
        - CPU operations are slower
- Graphics Processor (Graphics Card)

        - **Role**
            - Responsible for rendering images to display (computer screen, etc.)
            - Contains GPU (Graphical Processing Unit) and memory to handle all graphical operations
            - Does highly intensive graphical operations in parallel
        - **Effect if Limited Graphics Card Capability**
            - Graphical operations shifted to CPU, slowing it down
            - Can lead to reduced graphics quality
    - Network Connectivity (Network Interface)
        - **Types of Network Connectivity Devices**
            - NIC - Network Interface Card - Connects to ethernet cords
            - WNIC - Wireless Network Interface Card - Connects to wireless networks
            - Bluetooth - Communicates via bluetooth signals
            - 3G/4G/5G - Communicates via cellular standard radio waves
        - **Effects of Limited Network Connectivity**
            - Access to other computers over network is very slow
            - If network is internet-enabled, internet access can be very slow
- Types of Computing Devices
    - Mainframes
        - One computer with many multicore processors (100s of processors)
        - Vast amounts of RAM (100s of terabytes) and Secondary Storage (1000s of Petabytes)
        - Examples: Airline Reservation Systems, Payroll Processing, Weather/Financial Models and Prediction
    - Server Farms
        - Consists of many powerful computers connected and working in parallel
        - Each computer could have multiple processors, terabytes of secondary storage and 16 gigabytes of RAM or more
        - Examples: Data Centers, Cloud Hosting, Scientific Simulations/Predictions, 3D Rendering
- Operating System Roles
    - Peripherals - devices that can be connected to computer (mouse, keyboard, speakers, printer etc.), but are not essential
        - **Drivers** are software programs that translate the signal from the peripheral into digital data that can be interpreted by a computer
        - Majority of drivers are included as part of the OS, but can also be installed.
    - Memory Management
        - OS ensures that each program is allotted its own chunk (set of memory addresses) of RAM, known as **memory space**.
        - OS also ensures that programs doesn’t access or modify the memory space of other programs

    - OS manages the **virtual memory** feature by transferring **pages** of data from RAM to secondary storage when RAM is overloaded.
    - Data is transferred back to RAM from virtual memory, when there is free space.
- Paging
    - After placing data in secondary storage, as soon as there is empty space in the RAM, data will be transferred back.
    - Swapping is controlled by computer's **memory management unit** (MMU)
    - MMU may use any number of algorithms to choose pages to be swapped back to the RAM, including **Least Recently Used** (LRU), **Least Frequently Used** (LRU), and **Most Recently Used** (MRU)
    - File in secondary storage that contains pages is called the **pagefile** (or swapfile)
- Secondary Storage Management
    - OS provides folder (directory) structure for files in secondary storage
    - OS also manages security of these folders (user access rules)
- User Interface
    - Examples include, GUI, CLI, etc.
    - Used to interact with computers
    - Allows user to give commands to computer
    - Translates any input or output via the user interface and sends it to/from the correct memory address
    - Two types of OSes - **GUI-based** (Windows) and **CLI-based** (some versions of Linux)
- Time Slicing
    - Time slicing managed and coordinated by OS
    - A **time-slice** is the set amount of processing time a particular program or user is allotted for CPU usage
    - **Slices (called threads)** are scheduled for execution by the scheduler program.
    - Slices representing tasks from different programs can "take turns" being executed.
- Interrupts & Interrupt Handlers
    - Signal to processor emitted by hardware or software indicating an event that needs immediate attention
    - OS pauses current action, saving state, and initiating a program called an **interrupt handler** to deal with the high-priority event
    - After interrupter handler is finished, normal operations resume
    - Two types of interrupts: **hardware interrupts** and **software interrupts**
    - **Examples of Hardware Interrupts:** Printer paper jam, keyboard press by user, disk drive indicating it is ready for more data
    - **Examples of Software Interrupts:** Software error/malfunction, Infinite loop

    - being halted
- Polling
    - Process where one program or device repeatedly asks for the status of another so that it can execute some action
    - **Example**: A computer repeatedly (every 5 seconds) checks whether a printer is connected
    - Common alternative is interrupts
    - **Example**: When a printer is connected, an interrupt can be sent to the CPU to make the computer aware that it is connected

Interrupts	**Polling**
Interrupts CPU to execute action	Constantly checking another device to see if ready to execute action
Interrupt handler deals with interrupt	CPU, or another program processes status check
Requires very few CPU cycles	Wastes CPU cycles because of constant response-request actions
Inefficient if CPU is constantly interrupted	Inefficient if polling rarely results in action

- Scheduling
    - Act of assigning tasks to resources (processor, network interface, graphics card, etc.)
    - Handled by an OS-based process called a **scheduler**.
    - Allocates tasks to resources based on some algorithm (Round Robin, etc.)
    - Goal is to keep all resources busy, allow users to share resources effectively, and overall complete all tasks as quickly and efficiently as possible

Topic 7: Control Systems

- Control System - a device, or set of devices that controls, manages, commands, directs or regulates the behavior of other devices or systems
    - In the context of IBCS, some sensor or electrical input is processed by a microprocessor to generate some output
    - Examples: refrigerators, air conditioners, traffic lights (current time is input)

Advantages	**Disadvantages**
Computer can respond much more quickly than humans	Technical malfunctions can occur
Control systems can run without a break (24/365)	Cannot react to unexpected events
Control systems are less error prone than humans	Relies on consistent supply of electricity
Control systems can be placed in environments hazardous to humans	Could be more expensive than a human solution

- Control System Components
    - Microprocessor
        - an integrated circuit (chip) that contains all the functions of a computer's CPU (arithmetic, logical, and control operations)
        - accepts electrical signal, processes, and outputs different signal
        - Performs calculations and data processing
    - Sensor
        - a device which detects or measures a physical property and records, indicates, or otherwise responds to it
        - Examples:
            - **Heat**: measures temperature, e.g. central heating, fire alarm
            - **Humidity**: measures water vapor in the air, e.g. greenhouses, swimming pool halls
            - **Infrared**: measures radiation, e.g. security alarm systems
            - **Light (photodiode/phototransistor)**: brightness, e.g. security lights
            - **pH**: acidity levels, e.g. environmental monitoring
            - **Pressure (piezoresistive pressure sensor)**: force applied on the sensor, e.g. automatic doors, alarm systems
            - **Smoke**: particles in the air, e.g. fire alarm
            - **Sound**: sound pressure level, e.g. noise pollution monitoring, voice controlled systems, alarm systems

        - **Tilt**: angle of tilt, e.g. aircrafts, alarm systems installed in windows
        - **Touch**: more sensitive than pressure/detects contact, e.g. robots
  - Actuator
    - converts input energy of one form to another
    - In IBCS, input energy is usually an electrical signal that is converted to physical motion
    - requires a control device (which emits a control signal) and a source of energy
    - Examples:
      - **LED/light bulb**: creates light, e.g. display of information
      - **Heater**: increases temperature, e.g. central heating
      - Cooling unit: decreases temperature, e.g. central heating, AC
      - **Motor**: spins things around, e.g. robots, washing machines, elevator
      - **Pump**: pushes air/water through pipes, e.g water cleaning system, process control
      - **Buzzer/bell/siren**: creates (loud) noises, e.g. fire alarm
- Transducers
  - Confirms one form of energy into another
  - **Sensors** and **actuators** are both transducers
- **Analog-to-Digital (ADC) Converters** - Converts analog data from sensors into digital data, which can be read by computers
- IPO (Input-Process-Output) Model
  1) Sensors take in analog input and convert them to digital data.
  2) Digital data is processed by the microprocessor, to output different digital data.
  3) Transducers turn this digital data into some sort of physical motion.
- Feedback - If either the output or some form of the output is utilized as part of the system input, then it is known as feedback.
- Feedback Loops
  - **Open Loop System**
    - Doesn't take feedback into account, repeats same action over and over again based on input
    - Examples: TV Remote Control, Light Bulb, Coffee Machine, Toaster
  - **Closed Loop System**
    - Takes feedback into account, including output and possibly other factors external to the system
    - Correct errors in output by using feedback loop to alter subsequent inputs
    - Example: Home Thermostat, Airplane Autopilot
  - **Negative Feedback Loop**
    - A system in which feedback is given in order to reduce fluctuation in subsequent output
    - Try to move output closer to some equilibrium, or target value
  - **Positive Feedback Loop**

    - Enhances or amplified changes in output to move a system further away from equilibrium or a given value.

- **Centralized System** - All components are controlled by a centralized processing unit

Advantages	Disadvantages
Easier to maintain and troubleshoot	If main sensor/controller/actuator fails, the whole system fails
More control	Less power
	Less flexibility

- **Distributed System** - Different components are controlled by a number of different processing units located throughout the system - all work towards common goal

Advantages	Disadvantages
Shared processing load	Harder to maintain
More reliability	More complex software
Flexibility	

- Autonomous Agents
    - "Autonomous agents are software programs which respond to states and events in their environment independent from direct instruction by the user or owner of the agent, but acting on behalf and in the interest of the owner."
    - Examples: recommendation systems, drone warms, search engine crawlers
    - 4 Characteristics of Autonomous Agents
        1) **Autonomy** - Can selected independently select tasks in order to achieve overarching goal
        2) **Reactive** - Senses environment and reacts based on this input
        3) **Concurrency/Sociality** - Can interact with other agents cooperatively, competitively, or in coordination
        4) **Persistence** - an agent consistently acts in pursuit of its goal