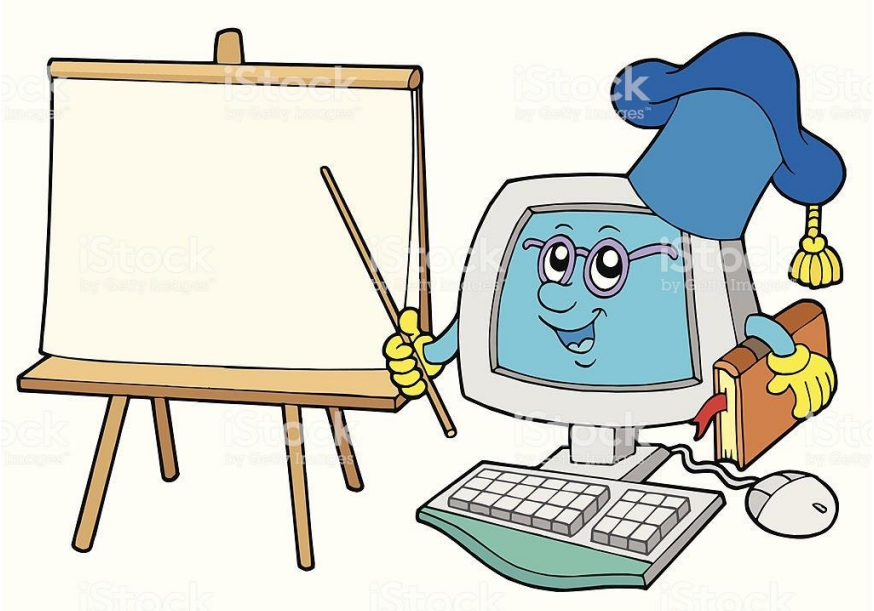
HIGHER SECONDARY SECOND YEAR

COMPUTER SCIENCE

UNIT I - Problem Solving Techniques

BOOK BACK QUESTION & ANSWERS

2024 - 25



Prepared By,

J. KAVITHA, B.Sc, B.Ed, M.C.A, M.Phil.,

Computer Instructor Gr - I

GHSS, S.S.KULAM

Coimbatore – 641107.

https://kavikalvi.freeweb.co.in/

CHAPTER 1: Function

Choose the best answer: (1 Mark)

1. The small sections of code that are used to perform a particular task is called
   (A) Subroutines (B) Files (C) Pseudo code (D) Modules
2. Which of the following is a unit of code that is often defined within a greater code structure?
   (A) Subroutines **(B) Function** (C) Files (D) Modules
3. Which of the following is a distinct syntactic block?
   (A) Subroutines (B) Function **(C) Definition** (D) Modules
4. The variables in a function definition are called as
   (A) Subroutines (B) Function (C) Definition **(D) Parameters**
5. The values which are passed to a function definition are called
   (A) Arguments (B) Subroutines (C) Function (D) Definition
6. Which of the following are mandatory to write the type annotations in the function definition?
   (A) { } **(B) ( )** (C) [ ] (D) < >
7. Which of the following defines what an object can do?
   (A) Operating System (B) Compiler **(C) Interface** (D) Interpreter
8. Which of the following carries out the instructions defined in the interface?
   (A) Operating System (B) Compiler
   (C) Implementation (D) Interpreter
9. The functions which will give exact result when same arguments are passed are called
   (A) Impure functions (B) Partial Functions
   (C) Dynamic Functions **(D) Pure functions**
10. The functions which cause side effects to the arguments passed are called
    (A) Impure functions (B) Partial Functions
    (C) Dynamic Functions (D) Pure functions

Answer the following questions: (2 Marks)

1. **What is a subroutine?**
   - Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly.
   - In Programming languages these subroutines are called as Functions.
2. **Define Function with respect to Programming language.**
   - A function is a unit of code that is often defined within a greater code structure.
   - A function works on many kinds of inputs and produces a concrete output
3. **Write the inference you get from X:=(78).**
   - **X:=(78)** is a function definition.
   - Definitions bind values to names. Hence, the value **78** bound to the name **"X"**.
4. **Differentiate interface and implementation.**

Interface	Implementation
Interface just defines what an object can do, but won't actually do it	Implementation carries out the instructions defined in the interface.

5. **Which of the following is a normal function definition and which is recursive function definition**

i) let sum x y:
  return x + y **Ans: Normal Function**

ii) let disp:
  print 'welcome' **Ans: Normal Function**

iii) let rec sum num:
  if (num!=0) then return num + sum (num-1)
  else return num **Ans: Recursive Function**

Answer the following questions: (3 Marks)

1. **Mention the characteristics of Interface.**
   - The class template specifies the interfaces to enable an object to be created and operated properly.
   - An object's attributes and behavior is controlled by sending functions to the object.
2. **Why strlen is called pure function?**
   - Each time we call the **strlen() function** with the same parameters, it always gives the same correct answer. So it is a pure function.

3. **What is the side effect of impure function? Give example.**
   - The variables used inside the function may cause side effects though the functions which are not passed with any arguments. In such cases the function is called impure function.
   - When a function depends on variables or functions outside of its definition block, you can never be sure that the function will behave the same every time it's called. **Example:** random() function.

4. **Differentiate pure and impure function.**

Pure function	Impure function
The return value of the pure functions solely depends on its arguments passed.	The return value of the impure functions does not solely depend on its arguments passed.
Pure functions will give exact result when the same arguments are passed.	Impure functions never assure you that the function will behave the same every time it's called.
They do not have any side effects.	They have side effects.
They do not modify the arguments which are passed to them.	They may modify the arguments which are passed to them.

5. **What happens if you modify a variable outside the function? Give an example.**
   - Modifying the variable outside of function causes side effect.
   - Example:

```
let y := 0
(int) inc (int)x
y := y+x;
return(y)
```

   - Here, the result of **inc()** will change every time if the value of **'y'** get changed inside the function definition.
   - Hence, the side effect of inc () function is changing the data of the external variable **'y'.**

Answer the following questions: (5Marks)

1. What are called Parameters and write a note on

(i) Parameter without Type (ii) Parameter with Type

- **Parameters** are the variables in a function definition.

(i) Parameter Without Type:

- **S**ome language compilers solve this data type problem algorithmically, if we do not specify the data types of variables in the function.

Example:

```
let rec pow a b:=
    if b=0 then 1
    else a * pow a (b-1)
```

- In the above function definition if expression can return **1** in the then branch, shows that as per the **typing** rule the entire if expression has type **int**.
- Since **'a'** is multiplied with another expression using the * operator, **'a'** must be an int.

(ii) Parameter with Type:

Example:

```
let rec pow (a: int) (b: int) : int :=
if b=0 then 1
else a * pow b (a-1)
```

- When we write the type annotations for **'a'** and **'b'** the parentheses are mandatory.
- Generally we can leave out these annotations, because it's simpler to let the compiler infer them.

2. Identify in the following program

```
let rec gcd a b :=
if b <> 0 then gcd b (a mod b) else return a
```

i) Name of the function - **gcd**

ii) Identify the statement which tells it is a recursive function - **let rec gcd a b :=**

iii) Name of the argument variable - **a, b**

iv) Statement which invoke the function recursively - **gcd b(a mod b)**

v) Statement which terminates the recursion - **return a**

3. Explain with example Pure and impure functions.

Pure functions :

- Pure functions are functions which will give exact result when the same arguments are passed.
- A function can be a pure function provided it should not have any external variable which will alter the behaviour of that variable.
- For example the mathematical function sin (0) always results **0**. This means that every time you call the function with the same arguments, you will always get the same result.

Impure functions:

- The variables used inside the function may cause side effects though the functions which are not passed with any arguments. In such cases the function is called impure function.
- When a function depends on variables or functions outside of its definition block, you can never be sure that the function will behave the same every time it's called.
- For example the mathematical function random() will give different outputs for the same function call.

4. Explain with an example interface and implementation.

- An interface is a set of action that an object can do.
- **For example**, when you press a light switch, the light goes on, you may not have cared how it splashed the light.
- Object Oriented Programming language, an Interface is a description of all functions.
- In object oriented programs classes are the interface and how the object is processed and executed is the implementation.

  Example, consider the following implementation of a function that finds the minimum of its three arguments:

```
let min 3 x y z :=
        if x < y then
                if x < z then x else z
        else
                if y < z then y else z
```

CHAPTER 2: Data Abstraction

Choose the best answer: (1 Mark)

1. Which of the following functions that build the abstract data type?
   (A) Constructors (B) Destructors (C) recursive (D) Nested
2. Which of the following functions that retrieve information from the data type?
   (A) Constructors **(B) Selectors** (C) recursive (D) Nested
3. The data structure which is a mutable ordered sequence of elements is called
   (A) Built in **(B) List** (C) Tuple (D) Derived data
4. A sequence of immutable objects is called
   (A) Built in (B) List **(C) Tuple** (D) Derived data
5. The data type whose representation is known are called
   (A) Built in datatype (B) Derived datatype
   (C) Concrete datatype (D) Abstract datatype
6. The data type whose representation is unknown are called
   (A) Built in datatype (B) Derived datatype
   (C) Concrete datatype **(D) Abstract datatype**
7. Which of the following is a compound structure?
   (A) Pair (B) Triplet (C) single (D) quadrat
8. Bundling two values together into one can be considered as
   (A) Pair (B) Triplet (C) single (D) quadrat
9. Which of the following allow to name the various parts of a multi-item object?
   (A) Tuples (B) Lists **(C) Classes** (D) quadrats
10. Which of the following is constructed by placing expressions within square brackets?
    (A) Tuples **(B) Lists** (C) Classes (D) quadrats

Answer the following questions: (2 Marks)

1. **What is abstract data type?**
   - Abstract Data type (ADT) is a type for objects or classes whose behavior is defined by a set of value and a set of operations.
2. **Differentiate constructors and selectors.**

CONSTRUCTORS	SELECTORS
Constructors are functions that build the abstract data type.	Selectors are functions that retrieve information from the data type.
Constructors create an object, bundling together different pieces of information	Selectors extract individual pieces of information from the object.

3. **What is a Pair? Give an example.**
   - Any way of bundling two values together into one can be considered as a pair.
   - Lists are a common method to do so. Therefore List can be called as Pairs.
     Example: lst[(0,10),(1,20)]
4. **What is a List? Give an example.**
   - List is constructed by placing expressions within square brackets separated by commas. List can store multiple values of any data type.
     Example: lst[10,20]
5. **What is a Tuple? Give an example.**
   - A tuple is a comma-separated sequence of values surrounded with parentheses.
     Example: Color= ('red', 'blue', 'Green')

Answer the following questions: (3 Marks)

1. **Differentiate Concrete data type and abstract data type.**

CONCRETE DATA TYPE	ABSTRACT DATA TYPE
Concrete data types or structures (CDT's) are direct implementations of a relatively simple concept.	Abstract Data Types (ADT's) offer a high level view of a concept independent of its implementation.
A concrete data type is a data type whose representation is known.	Abstract data type the representation of a data type is unknown.

2. **Which strategy is used for program designing? Define that Strategy.**
   - A powerful strategy for designing programs is **'wishful thinking'**.
   - Wishful Thinking is the formation of beliefs and making decisions according to what might be pleasing to imagine instead of by appealing to reality.

3. **Identify Which of the following are constructors and selectors?**

(a) N1=number()	-	**Constructor**
(b) accetnum(n1)	-	**Selector**
(c) displaynum(n1)	-	**Selector**
(d) eval(a/b)	-	**Selector**
(e) x,y= makeslope (m), makeslope(n)	-	**Constructor**
(f) display()	-	**Selector**

4. **What are the different ways to access the elements of a list. Give example.**

The elements of a list can be accessed in two ways.

1.Multiple Assignment:

- Which unpacks a list into its elements and binds each element to a different name.

Example: lst := [10, 20] x, y := lst

x will become10 and ***y*** will become 20.

2.Element Selection Operator:

- It is expressed using square brackets.
- A second method for accessing the elements in a list is by the element selection operator.

Example: lst[0] 10 lst[1] 20

5. **Identify Which of the following are List, Tuple and class ?**

(a) arr [1, 2, 34]	-	**List**
(b) arr (1, 2, 34)	-	**Tuple**
(c) student [rno, name, mark]	-	**Class**
(d) day= („sun‟, „mon‟, „tue‟, „wed‟)	-	**Tuple**
(e) x= [2, 5, 6.5, [5, 6], 8.2]	-	**List**
(f) employee [eno, ename, esal, eaddress]	-	**Class**

Answer the following questions: (5Marks)

1. **How will you facilitate data abstraction. Explain it with suitable example.**

To facilitate data abstraction, you will need to create constructors and selectors.

- Constructors are functions that build the abstract data type.
- Selectors are functions that retrieve information from the data type.

For example,

Let's take an abstract datat ype called city. This city object will hold the city's name, and its latitude and longitude.

city := makecity (name, lat, lon)

- Here the function makecity (name, lat, lon) is the constructor. When it creates an object city, the values name, lat and lon are sent as parameters.
- getname(city), getlat(city) and getlon(city) are selector functions that obtain information from the object city.

2. What is a List? Why List can be called as Pairs. Explain with suitable example.

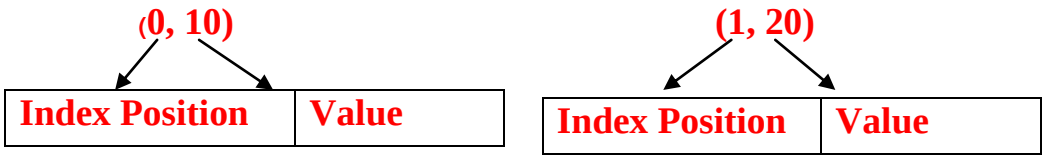
LIST:

- List is constructed by placing expressions within square brackets separated by commas. List can store multiple values of any data type.

PAIR:

- Any way of bundling two values together into one can be considered as a pair. Lists are a common method to do so. Therefore List can be called as Pairs.

  Example:lst[(0,10),(1,20)]



3. How will you access the multi-item. Explain with example.

- List does not allow naming the various parts of a multi-item object.
- Instead of using a list, you can use the structure construct (In OOP languages it's called class construct) to represent multi-part objects where each part is named.

  Example:

```
class Person:                main()
person( )                    p1:=Person()
firstName := " “             firstName := " Padmashri “
id := " “                    id :="994-222-1234“
email := " “                 email="compsci@gmail.com"
```

- Same way using class you can create many objects of that type.

CHAPTER 3: Scoping

Choose the best answer: (1 Mark)

1. Which of the following refers to the visibility of variables in one part of a program to another partof the same program.
   (A) Scope (B) Memory (C) Address (D) Accessibility
2. The process of binding a variable name with an object is called
   (A) Scope **(B) Mapping** (C) late binding (D) early binding
3. Which of the following is used in programming languages to map the variable and object?
   (A):: (B) := **(C) =** (D) ==
4. Containers for mapping names of variables to objects is called
   (A) Scope (B) Mapping (C) Binding **(D) Namespaces**
5. Which scope refers to variables defined in current function?
   (A) Local Scope (B) Global scope (C) Module scope (D) Function Scope
6. The process of subdividing a computer program into separate sub-programs is called
   (A) Procedural Programming **(B) Modular programming**
   (C) Event Driven Programming (D) Object oriented Programming
7. Which of the following security technique that regulates who can use resources in a computingenvironment?
   (A) Password (B) Authentication
   (C) Access control (D) Certification
8. Which of the following members of a class can be handled only from within the class?
   (A) Public members (B) Protected members
   (C) Secured members **(D) Private members**
9. Which members are accessible from outside the class?
   (A) Public members (B) Protected members
   (C) Secured members (D) Private members
10. The members that are accessible from within the class and are also available to its subclasses iscalled
    (A) Public members **(B) Protected members**
    (C) Secured members (D) Private members

Answer the following questions: (2 Marks)

1. **What is a scope?**
   - Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.
2. **Why scope should be used for variable. State the reason.**
   - The scope should be used for variables because; it limits a variable's scope to a single definition.
   - That is the variables are visible only to that part of the code.
3. **What is Mapping?**
   - The process of binding a variable name with an object is called mapping.
   - = (equal to sign) is used in programming languages to map the variable and object.
4. **What do you mean by Namespaces?**
   - Namespaces are containers for mapping names of variables to objects.
     Example: a = 5, Here the variable 'a' is mapped to the value '5'.
5. **How Python represents the private and protected Access specifiers?**
   - Python prescribes a convention of adding a prefix **__(double underscore)** results in a variable name or method becoming **private**.
     Example: self.__n2 = n2
   - Adding a prefix **_(single underscore)** to a variable name or method makes it protected. **Example: self._sal = sal**

Answer the following questions: (3 Marks)

1. **Define Local scope with an example.**
   - Local scope refers to variables defined in current function.
   - A function will always look up for a variable name in its local scope.
   - Only if it does not find it there, the outer scopes are checked.

   Example:

```
a = 10
def Inner():
    a = 20
    print(a)
Inner()
```

   Output: 20

2. **Define Global scope with an example.**
   - A variable which is declared outside of all the functions in a program is known as global variable.
   - Global variable can be accessed inside or outside of all the functions in a program.

   Example:

```
a = 10
def inner():
        a = 20
        print(a)
inner()
print(a)
```

   Output:

```
20
10
```

3. Define Enclosed scope with an example.

- A variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.

Example:

```
def outer():
    b = 10
    def inner():
        a = 20
        print(a)
        print(b)
    inner()
outer()
```

Output:

```
20
10
```

4. Why access control is required?

- Access control is a security technique that regulates who or what can view or use resources in a computing environment.
- It is a fundamental concept in security that minimizes risk to the object.
- In OOPS Access control is implemented through access modifiers.

5. Identify the scope of the variables in the following pseudo code and write its output

```
color:= 'Red'              - Global
mycolor():
b:='Blue'                  - Enclosed
myfavcolor():
g:='Green'                 - Local
print color, b, g
myfavcolor()
print color, b
mycolor()
print color
```

OUTPUT:

```
Red Blue Green
Red Blue
Red
```

Answer the following questions: (5 Marks)

1. Explain the types of scopes for variable or LEGB rule with example.

- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.
- The **LEGB** rule is used to decide the order in which the scopes are to be searched for scope resolution.

TYPES OF VARIABLE SCOPE:

Local(**L**)	Defined inside function/class
Enclosed(**E**)	Defined inside enclosing functions (Nested function concept)
Global(**G**)	Defined at the uppermost level
Built-in (**B**)	Reserved names in built-in functions (modules)

(i) Local scope:

- Local scope refers to variables defined in current function.
- A function will always look up for a variable name in its local scope.
- Only if it does not find it there, the outer scopes are checked.

(ii) Enclosed scope with an example.

- A variable which is declared inside a function which contains another function definition with init, the inner function can also access the variable of the outer function. This scope is called enclosed scope.

(iii) Global scope:

- A variable which is declared outside of all the functions in a program is known as global variable. Global variable can be accessed inside or outside of all the functions in a program.

(iv) Built-In-Scope:

- The built-in scope has all the names that are pre-loaded into the program scope when we start the compiler or interpreter.
- Any variable or module which is defined in the library functions of a programming language has Built-in or module scope.

Example:

```
x = 10
z = 30                                  #Global
def outer():
        y = 20                                  #Enclosed
        def inner():
                x = 40                          #Local
                print(f'x is {x}')
                print(f'y is {y}')
                print(f'z is {z}')
                print(len("abc"))       #Built-in
         inner()
outer()
```

Output:

```
x is 40
y is 20
z is 30
3
```

2. **Write any Five Characteristics of Modules.**
   - Modules contain instructions, processing logic, and data.
   - Modules can be separately compiled and stored in a library.
   - Modules can be included in a program.
   - Module segments can be used by invoking a name and some parameters.
   - Module segments can be used by other modules.
3. **Write any five benefits in using modular programming.**
   - Less code to be written.
   - A single procedure can be developed for reuse.
   - Programs can be designed easily because a small team deals with only a small part of the entire code.
   - The code is stored across multiple files.
   - Code is short, simple and easy to understand.
   - Errors can easily be identified, as they are localized to a subroutine or function.
   - The same code can be used in many applications.
   - The scoping of variables can easily be controlled.

CHAPTER 4: Algorithmic Strategies

Choose the best answer: (1 Mark)

1. The word comes from the name of a Persian mathematician Abu Ja'far Mohammed ibn-i Musa alKhowarizmi is called?
   (A) Flowchart (B) Flow **(C) Algorithm** (D) Syntax
2. From the following sorting algorithms which algorithm needs the minimum number of swaps?
   (A) Bubble sort (B) Quick sort (C) Merge sort **(D) Selection sort**
3. Two main measures for the efficiency of an algorithm are
   (A) Processor and memory (B) Complexity and capacity
   (C) Time and space (D) Data and space
4. The algorithm that yields expected output for a valid input in called as
   (A) Algorithmic solution (B) Algorithmic outcomes
   (C) Algorithmic problem (D) Algorithmic coding
5. Which of the following is used to describe the worst case of an algorithm?
   (A) Big A (B) Big S (C) Big W **(D) Big O**
6. Big Ω is the reverse of
   (A) Big O (B) Big θ (C) Big A (D) Big S
7. Binary search is also called as
   (A) Linear search (B) Sequential search
   (C) Random search **(D) Half-interval search**
8. The Θ notation in asymptotic evaluation represents
   (A) Base case **(B) Average case** (C) Worst case (D) NULL case
9. If a problem can be broken into subproblems which are reused several times, the problempossesses which property?
   (A) Overlapping subproblems (B) Optimal substructure
   (C) Memoization (D) Greedy
10. In dynamic programming, the technique of storing the previously calculated values is called?
    (A) Saving value property (B) Storing value property
    (C) Memoization (D) Mapping

Answer the following questions: (2 Marks)

1. **What is an Algorithm?**
   - An algorithm is a finite set of instructions to accomplish a particular task.
   - It is a step-by-step procedure for solving a given problem**.**
2. **Write the phases of performance evaluation of an algorithm.**
   - **A Priori estimates:** This is a theoretical performance analysis of an algorithm. Efficiency of an algorithm is measured by assuming the external factors.
   - **A Posteriori testing:** This is called performance measurement. In this analysis, actual statistics like running time and required for the algorithm executions are collected.
3. **What is Insertion sort?**
   - Insertion sort is a simple sorting algorithm. It works by taking elements from the list one by one and inserting then in their correct position in to a new sorted list.
4. **What is Sorting?**
   - Sorting is a process of arranging group of items in an ascending or descending order.

     Types: Bubble Sort, Selection Sort, Insertion sort.
5. **What is searching? Write its types.**
   - A Search algorithm is the step-by-step procedure used to locate specific data among a collection of data.

     Types: Linear Search, Binary Search

Answer the following questions: (3 Marks)

1. **List the characteristics of an algorithm.**
   - Input * Output * Finiteness * Definiten
   - Effectiveness * Correctness * Simplicity * Unambiguous
   - Feasibility * Portable * Independent
2. **Discuss about Algorithmic complexity and its types.**
   - The complexity of an algorithm f(n) gives the running time and/or the storage space required by the algorithm in terms of n as the size of input data.

Types of complexity:

1. Time Complexity: The Time complexity of an algorithm is given by the number of steps taken by the algorithm to complete the process.

2. Space Complexity: Space complexity of an algorithm is the amount of memory required to run to its completion.

3. **What are the factors that influence time and space complexity.**

The two main factors, which decide the efficiency of an algorithm are,

- **Time Factor** -Time is measured by counting the number of key operations like comparisons in the sorting algorithm.
- **Space Factor** - Space is measured by the maximum memory space required by the algorithm.

4. **Write a note on Asymptotic notation.**

- Asymptotic Notations are languages that use meaningful statements about time and space complexity.

  Big O - Worst-case of an algorithm.

  Big Ω - Best -case of an algorithm

  Big Θ - complexity case of an algorithm (Or) lower bound = upper bound

5. **What do you understand by Dynamic programming?**

- Dynamic programming is used when the solution to a problem can be viewed as the result of a sequence of decisions.
- The given problem will be divided into smaller overlapping sub-problems.
- An optimum solution for the given problem can be achieved by using result of smaller sub-problem.
- Dynamic algorithm uses Memorization.

Answer the following questions: (5 Marks)

1. **Explain the characteristics of an algorithm.**

Input	Zero or more quantities to be supplied.
Output	At least one quantity is produced.
Finiteness	Algorithms must terminate after finite number of steps.
Definiteness	All operations should be well defined. For example operations involving division by zero or taking square root for negative number are unacceptable.
Effectiveness	Every instruction must be carried out effectively.
Correctness	The algorithms should be error free.
Simplicity	Easy to implement.
Unambiguous	Algorithm should be clear and unambiguous. Each of its steps and their inputs/outputs should be clear and must lead to only one meaning.
Feasibility	Should be feasible with the available resources.
Portable	An algorithm should be generic, independent of any programming language or an operating system able to handle all range of inputs.
Independent	An algorithm should have step-by-step directions, which should be independent of any programming code.

2. Discuss about Linear search algorithm.

- Linear search also called sequential search is a sequential method for finding a particular value in a list.
- In this searching algorithm, list need not be ordered.

Pseudo code:

- Traverse the array using for loop.
- In every iteration, compare the target search key value with the current value of the list.
  - If the values match, display the current index and value of the array
  - If the values do not match, move on to the next array element.
  - If no match is found, display the search element not found.
- If no match is found, display the search element not found.

Example:

1. Input: values[] = {5, 34, 65, 12, 77, 35}
   target = 77
   Output: 4
2. Input: values[] = {101, 392, 1, 54, 32, 22, 90, 93}
   target = 200
   Output: -1 (not found)

3. What is Binary search? Discuss with example.

- Binary search also called half-interval search algorithm.
- It finds the position of a search element within a sorted array.

Example:

Let us assume that the **search element is 60** and we need to search the index of search element 60 using binary search.

- First, we find index of middle element by using this formula :
  mid = (low + high) / 2 , Here it is, (0 + 9) / 2 = 4.
- Compare the value stored at index 4 with target value, which is not match with search element. As the search value 60 >50.
- Now we change our search range **low to mid + 1** and find the new mid value as index 7.
- We compare the value stored at index 7 with our target value.
- Element not found because the value in index 7 is greater than search value. (80 > 60)
- Now we change our search range **low to mid - 1** and find the new mid value as index 5.
- We compare the value stored at location 5 with our search element.
- We found that it is a match.
- We can conclude that the search element 60 is found at location or index 5.
- If **no match is found** for all comparisons, then return -1.

4. Explain the Bubble sort algorithm with example.

- Bubble sort is a simple sorting algorithm; it starts at the beginning of the list of values stored in an array.
- It compares each pair of adjacent elements and swaps them if they are in the unsorted order.
- This comparison and passed to be continued until no swaps are needed, which shows the values in an array is sorted.

Pseudo code:

- Start with the first element i.e., index = 0, compare the current element with the next element of the array.
- If the current element is greater than the next element of the array, swap them.
- If the current element is less than the next or right side of the element, move to the next element.
- Go to Step 1 and repeat until end of the index is reached.

Example: Consider an array with values {15, 11, 16, 12, 14, 13}.

15 > 11 So Interchange	15	11	16	12	14	13
15 > 16 No Swapping	11	15	16	12	14	13
16 > 12 So Interchange	11	15	16	12	14	13
16 > 14 So Interchange	11	15	12	16	14	13
16 > 13 So Interchange	11	15	12	14	16	13

11	15	12	14	13	16

- The above pictorial example is for iteration-1.
- Similarly, remaining iteration can be done.
- At the end of all the iterations we will get the sorted values in an array as given below:

11	12	13	14	15	16

5. Explain the concept of Dynamic programming with suitable example.

- Dynamic programming is used when the solution to a problem can be viewed as the result of a sequence of decisions.
- Dynamic programming approach is similar to divide and conquer (i.e) the problem can be divided into smaller sub-problems.
- Results of the sub-problems can be re-used to complete the process.
- Dynamic programming approaches are used to find the solution in optimized way.

Steps to do Dynamic programming:

- The given problem will be divided into smaller overlapping sub-problems.
- An optimum solution for the given problem can be achieved by using result of smaller sub-problem.
- Dynamic algorithms uses Memoization.

Example: Fibonacci Iterative Algorithm with Dynamic Programming Approach

- Initialize f0=0, f1 =1
  step-1: Print the initial values of Fibonacci f0 and f1
  step-2: Calculate fibanocci fib ← f0 + f1
  step-3: Assign f0← f1, f1← fib
  step-4: Print the next consecutive value of fibanocci fib
  step-5: Go to step-2 and repeat until the specified number of terms generated
- For example if we generate fibonacci series up to 10 digits, the algorithm will generate the series as shown below:
  The Fibonacci series is : 0 1 1 2 3 5 8 13 21 34 55

Education is not just about going to school and getting a degree. It's about widening your knowledge and absorbing the truth about life.

ALL THE BEST!

J. KAVITHA, B.Sc, B.Ed, M.C.A, M.Phil.,
Computer Instructor Gr - I
GHSS, S.S.KULAM
Coimbatore – 641107.
✆: 8940762362