BCE Mid Sem - 1

Unit-2

Q1> Write an Algorithm to find the greater number between two numbers.

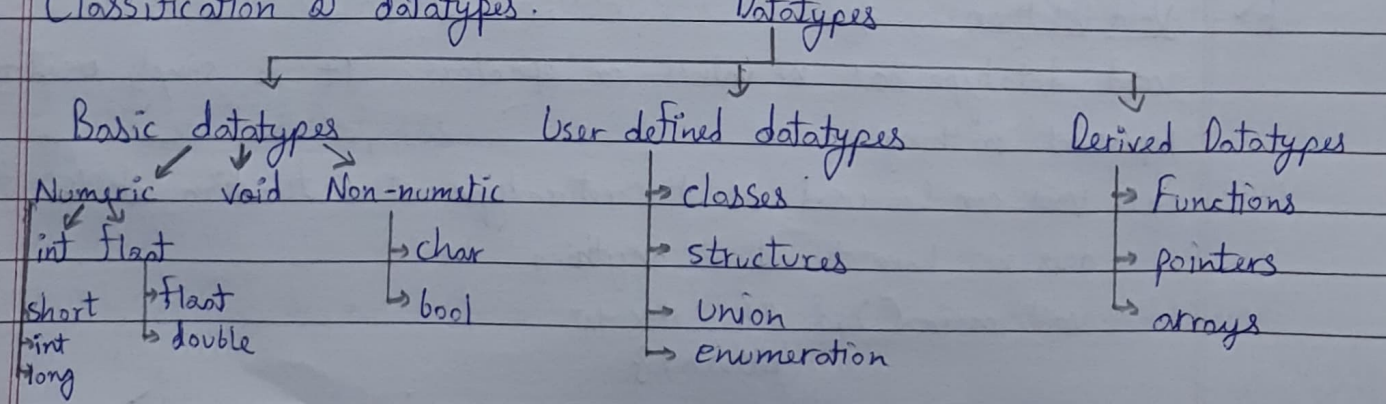
Ans
```
# include <iostream>
using namespace std;

int main(void) {
  int a, b;
  cout << "Enter first number = " << endl;
  cin >> a;
  cout << "Enter second number = " << endl;
  cin >> b;

  if (a>b) { cout << "Greater number is " << a; }
  else { cout << "Greater number is " << b; }

  return 0; }
```

output:-
```
Enter first number =
15
Enter second number =
3
Greater number is 3
```

Q2> Explain various Data types.

Ans Datatypes define the size and type of the variable to be used in a program.

Classification of datatypes:



(1) Basic datatypes

a) Numeric Datatypes

i) **int** : It is used to store integer values

* short → syntax → short varname ;
It stores values from -32768 to 32767 in 2bytes (16 bits) for all operating systems

* int → syntax → int varname ;
It stores values from -2147483648 to 2147483647 in 4 bytes for windows (32 bit or more) systems but in DOS system it only stores values from -32768 to 32767 in 2bytes

* long → Syntax → long varname ;
It stores values from 0 to 4294967296 in 4 bytes.
[unsigned]

ii) **float** : It is used to store decimal values.

* float → syntax → float varname = 3.14 ;
It ~~stores~~ takes 4 bytes in memory.

* double → syntax → double varname ;
It takes 8 bytes in the memory as it is used to store bigger decimal numbers than the range of float.

b) Void Datatype

void datatype has no values or operators. It is simply used to represent nothing.

eg → Void can be used as the return type of a function if the function does not have to return anything i.e

```
void main () { // line of codes
                return ; }
```

c) Non numeric datatypes

i) Char → Syntax → char varname = 'A';
It is used to store a single character, which is enclosed within single inverted commas, in ASCII codes.
It takes 1 byte in memory.

ii) bool → Syntax → bool b1 = true;
It is used to store either true or false.

(2) User defined datatypes

i) Classes
They are user defined datatypes that contain some data variables and their associated functions.
They are a group of similar objects.

ii) Structure
Structure is a datatype which can contain a variety of datatypes in it.

iii) Union
Union is a memory location that is shared by two or more different variables at different times.

iv) Enumeration
It is an alternative method for naming integer constants.

(3) Derived datatypes

i) Array
An array is a container object that holds a fixed number of

values of a single type

ii) Function

It is a derived datatype which is built for special purpose and with some standard definition.

iii) Pointer

It is a special type of variable which holds the address of another variable.

Syntax → int *ip;

ip = &x

Q 3) What is Operator? define different type of operators.

Ans Operators are special symbols that perform mathematical operations or logical operations on one, two or three operands and then return a result.

The Various types of operators

1) Arithmatic operators

These include : plus + → used for addition

minus − → used for subtraction

asterisk * → used for multiplication

forward slash / → used for division but returns an integer quotient

modulus % → used for finding the remainder after division.

2) Relational operators

These are used for comparision of two values and therefore used as conditions so as to return boolean value.

These include == (equals), != (not equal), < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to)

3> Assignment operator

It includes = which is used to assign value of right operand to the left operand.

4> Increment / decrement operators

These include prefix operators (++a, --a) and postfix operators (a++, a--)

These are unary operators

5> Logical operators

These operators are used which conditions to get a boolean in return

These include ! (not) && (and) || (or)

6> Conditional operator

This is a ternary operator

syntax → condition ? statement 1 : statement 2

It executes statement 1 if the condition is true or statement 2 if condition is false.

7> Compound operator

These include +=, -=, /=, %=, *=,

These are used if the same variable comes on both sides of assignment operator.

eg→ a = a+1 can be written as a+=1.

8> Bitwise operators

These include & (and), | (or), ^ (XOR), << (shift left), >> (shift right), ~ (one's complement)

These are used to operate on bits.

9> sizeof() operator

This is a unary operator which is used to find the size of the operand in bytes

Syntax → sizeof(type)
or sizeof(expression)

eg → sizeof(int) // 4
sizeof(5+8) // 4

Q4> What is variable? Explain different types of variables with suitable example.

Ans A variable is an identifier which identifies a certain part of the memory location and holds some specific type of information. In other works, variable is name given to some part of memory location.

* On basis of datatypes, types of variables are as follows:

bool → Stores either true or false.
char → stores a single character.
int → stores a integer value
float → stores a decimal value
double → stores a bigger decimal value than float.
void → represents absence of type

Syntax → datatype variable_names ;

eg → bool b1 = true ;
char a = 'A', oper = '+' ;
int num1 = 3, num2 = 4, num3 = 5 ;
float pi = 3.14 ;
double h = 2.6 7 8 9 10 11 12 13 14 ;
~~void~~

* On the basis of scope, types of variable are:

local variable → Variables that are defined within a function or block are said to be local to those functions
→ These variables do not exist outside that block so they can not be accessed or used outside the block

Global variable → Variables that are declared at the top of the program outside all of the functions or blocks are called global variables
→ These variables can be accessed from any part of the program throughout the lifetime of the program.

eg →
```
# include <iostream>
using name space std;

int g = 123;  // global variable
void func() {
        int l = 456;  // local variable
        cout << "local variable is " << l << endl; }
int main() { func();
        cout << "Global variable is " << g << endl;
return 0; }
```

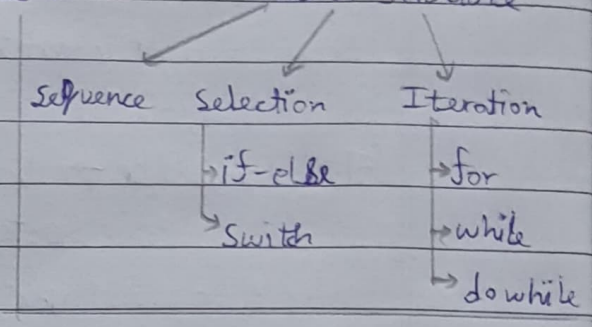
output →
```
local variable is 456
Global variable is 123
```

Here, g is the global variable and l is the local variable.

Q5> Explain different types of Control structure with examples.

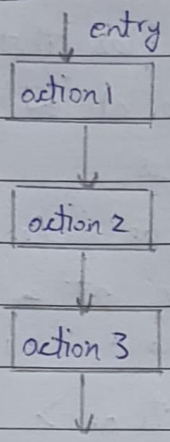
Ans Control structures are a way to specify the flow of control in program.

There are three types of control structures:



1> Sequence Structure (straight line)

In this type of structure a program executes line by line, as it is, in a sequence



Sequence structure

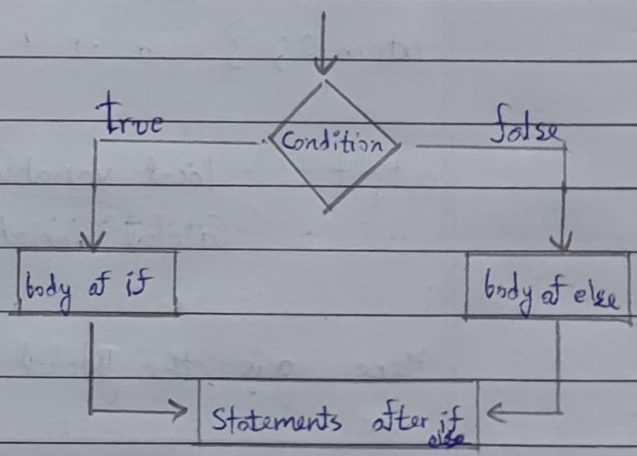
eg → # include <iostream>
using namespace std;

```
int main () { int num;
    cout << " Enter a number = ";
    cin >> num;
    cout << num << endl;
    cout << num +1 << endl;
    cout << num +2 << endl;
    return 0; }
```

output→ Enter a number = 1
1
2
3

2> Selection Structure (branching)

In selection structure, it involves a number of conditions or parameters on the basis of which the branching of flow of program takes place.



Selection structure

Eg→ # include <iostream>
using namespace std;

int main () { int num; bool greater = false;

```
cout << "Enter a number = ";
cin >> num ;

if (num < 5) { cout << num << "\n" << num+1 << "\n" << num+2 << endl ;}
else { cout<<num << "\n" << num +2 << "\n" << num +4 << endl ;
    greater = true ; }

Switch (greater) {
    case (true) :
        cout << " The number is greater than 5 " ;
        break ;
    default :
        cout << " The number is less than 5 " ; }
return 0 ;}
```

output → Enter a number = 2

2
3
4

The number is less than 5

OR

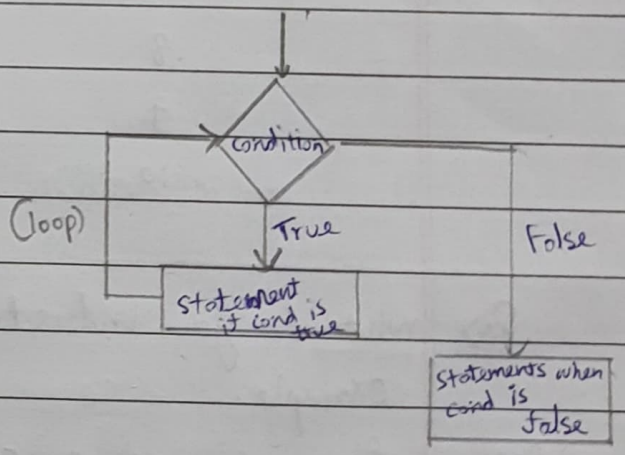
Enter a number = 10

10
12
14

The number is greater than 5

3> Looping / Iteration structure (repeatation)

In iteration control structure, a set of instructions can repeated until a certain condition is fulfilled.



Looping Structure.

eg →
```
# include <iostream>
using namespace std;

int main ( ) { int num1, num2 ;
cout << " Enter a number = " ;
cin >> num1 ;
```

```
cout << "Enter another number = ";
cin >> num2;

for (int i= num1; i>=0 ; i--) {
    cout << i << endl;
}

Cout<<" * * * * " << endl;

while (num2 <= 10) {
    cout << num2 << endl;
    num2 ++;
     }
return 0; }
```

output → Enter a number = 3
Enter another number = 8
3
2
1
0
* * * *
8
9
10

Q6) What do you understand by program structure give suitable example.

Ans Programs are a sequence of instructions or statements and the structure of these sequence of instructions is known as a program structure.

Example of a C++ program,

```
// This is a comment        → comment
# include <iostream>        → header file
using namespace std;        → using directive

int main () {               → main function
    Cout << " Hello World"; } statements
    return 0;
}
```

The program structure of the above program is as follows:

① Comments

* Comments, in a program, are used to explain code and make it more readable
* They can also be used to prevent execution of certain code when testing alternative code.
* They are are of two types : i) Single line comment
  eg → // This is a single line comment
  ii) Multi line comment, eg → /* This is a multiline comment */
* Compiler takes comments as single white space.

② Header files

* Header files contain a set of predefined standard library functions.
* They are accessed by using # include preprocessor directive.
* eg → iostream is a header file which contains functions like cin, cout to provide an input output flow.

③ Using directive

* Using directive is used to bring a specific member from the namespace into the current scope.

eg→ using namespace std; allows us to use cin, cout, endl directly as opposed to writing them as std::cin, std::cout std::endl in the absence of using directive.

④ Main function

* A main() function is a function that is called by the operating System at the program startup
It is the disgnated entry point to a program.

* Syntax → returntype function_name (arguments) { body }
eg → int main (void) { // statements
return 0; }

* The function body is defined within curly braces { } which contains programming statements within, each terminating with a semicolon (;).

⑤ Cin and cout Statements

* cout is output stream, used to display something on the screen
syntax → cout << output ; eg→ cout << "Hello world";
Here, << is the insertion operator

* Cin is input stream, used to take input from the user and store it in a variable.
syntax → Cin >> variable ; eg→ int a;
Here, >> is extraction operator Cin >> a;

⑥ String

* string is a sequence of character enclosed within double quotes
eg→ "Hello world"

⑦ return Keyword

The return statement returns the flow of execution to function from

where it is called, and terminates the flow of the current function.

Syntax → return expression_based_on_returntype ;

eg → int main ( ){ // code

return 0; }

Here int is the return type and after the execution of the return statement, flow of control is given back to the operating system since it was the return in main() function.

Q 7> Write a Program to Read and write 5 elements in an Array.

Sol

```
# include <iostream>
using namespace std;

int main ( ) { int my_array [5];

for (int i=0; i<5 ; i++){
cout << " element " << i+1 << " is ? " ;
cin >> my_array [i] ; }
```

```
for (int k=0 ; k<5 ; k++) {
    cout << " The array is " << my_array[k];
    }
return 0; }
```

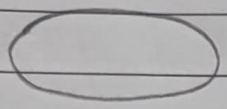
output → element 1 is ? 10
element 2 is ? 20
element 3 is ? 30
element 4 is ? 40
element 5 is ? 50
The array is 10 20 30 40 50

Q 8> What is flow chart and explain different symbols used in it.

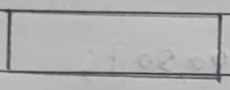
Ans Any algorithm or program is graphically represented by using flowcharts which are drawn using different symbols to represent different program constructs, and the lines connecting them show the flow of control.

Thus, a flow chart can be defined as the pictorial representation of an algorithm.

Commonly used flow chart symbols are :-

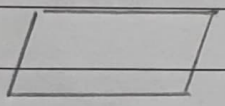


→ It is used to represent the start or stop of program

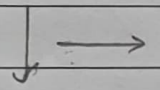


→ It is used to represent steps or operations performed in the program.

→ It is used to represent condition / branching statements, there is only on entry point but more than one exit point.



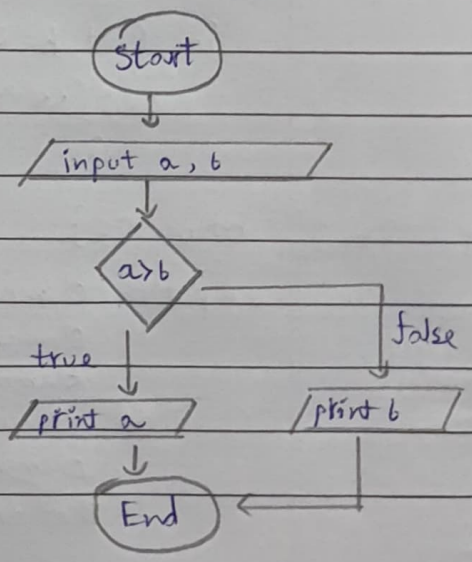
→ It is used to represents actions, like input, output operations, performed in the program



→ It is used to represent the flow of control.

eg → // program to print greater number

```
# include <iostream>
using namespace std;
int main () { int a, b;
    cin>>a ; cin>>b ;
if (a>b) { cout << "greater is " << a ;}
else { cout << "greater is " << b ;}
return 0 ; }
```



Q9> Write a program to illustrate Arithemstic operators.

Ans
```
// program to illustrate arithemstic operators
# include <iostream>
using namespace std;

int main ( ) { int a, b ; output ;
                char oper ;
cout << "Enter first number = "; cin >> a;
cout << "Enter second number = "; cin >> b;
cout << " Enter an operator = "; cin >> oper;

Switch (oper) {
   case ('+'):
                output = a+b;
                break;
   case ('-'):
                output = a-b;
                break;
   case ('*'):
                output = a*b;
                break;
   case ('/'):
                output = a/b;
                break;
   case ('%'):
                output= a%b;
                break;
   default:
        cout << "operator not identified"; }
cout << "a" << oper << "b =" << output;
return 0; }
```

output → Enter first number = 3
Enter second number = 2
Enter an operator = +
a+b = 5

→ Enter first number = 7
Enter Second number = 4
Enter an operator = -
a - b = 3

→ Enter first number = 5
Enter Second number = 10
Enter an operator = *
a * b = 50

→ Enter first number = 5
Enter Second number = 3
Enter an operator = /
a / b = 1

→ Enter first number = 4
Enter Second number = 2
Enter an operator = %
a % b = 0

→ Enter first number = 1
Enter second number = 2
Enter an operator = abc
operator not identified

Q10) Explain concept's of Object Oriented Programming.

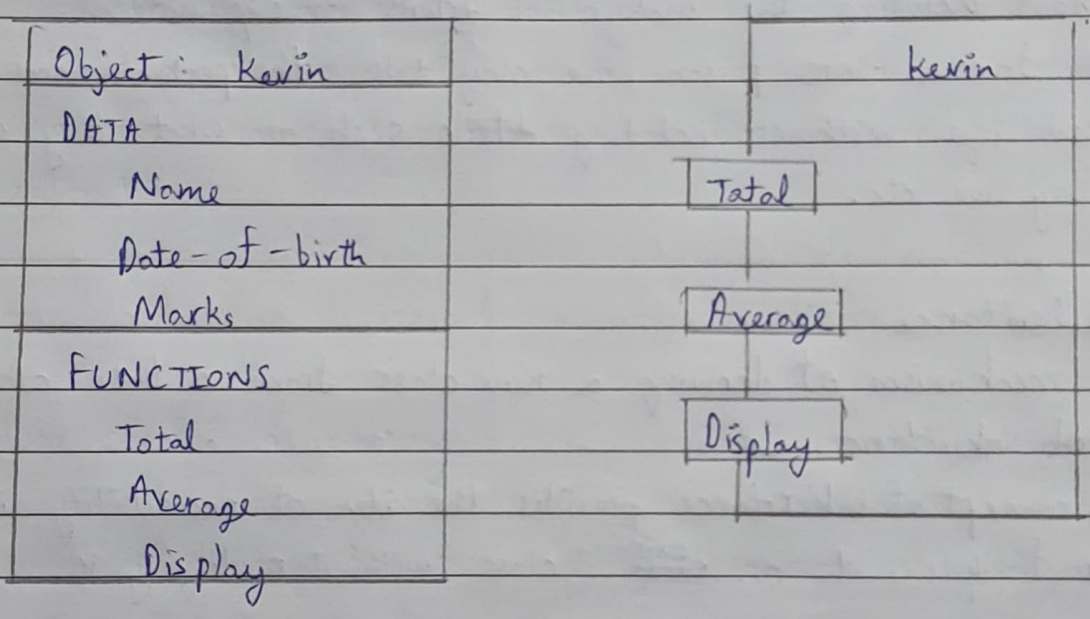
Ans Concepts of Object oriented Programming include:

1) Objects
2) Classes
3) Data abstraction and encapsulation
4) Inheritance
5) Polymorphism.

① Objects

Objects are the basic run-time entities in an object-oriented system having some characteristic and behaviour.

eg → Kevin is an object, of class Student, having characteristic i.e data such as name, date of birth, marks and behaviour ~~such~~ (functions) such as Total, Average, Display



Two ways of representing an object

② Classes

* A class can be defined as a user-defined datatype which contains data members and member functions to operate on those data members.
* It is a collection of similar objects.
* We can create any number of objects belonging to that class.

eg → int a, b, c;
fruit mango, apple, orange;

Here int, fruit are classes and a,b,c and mango, apple, orange are their objects respectively.

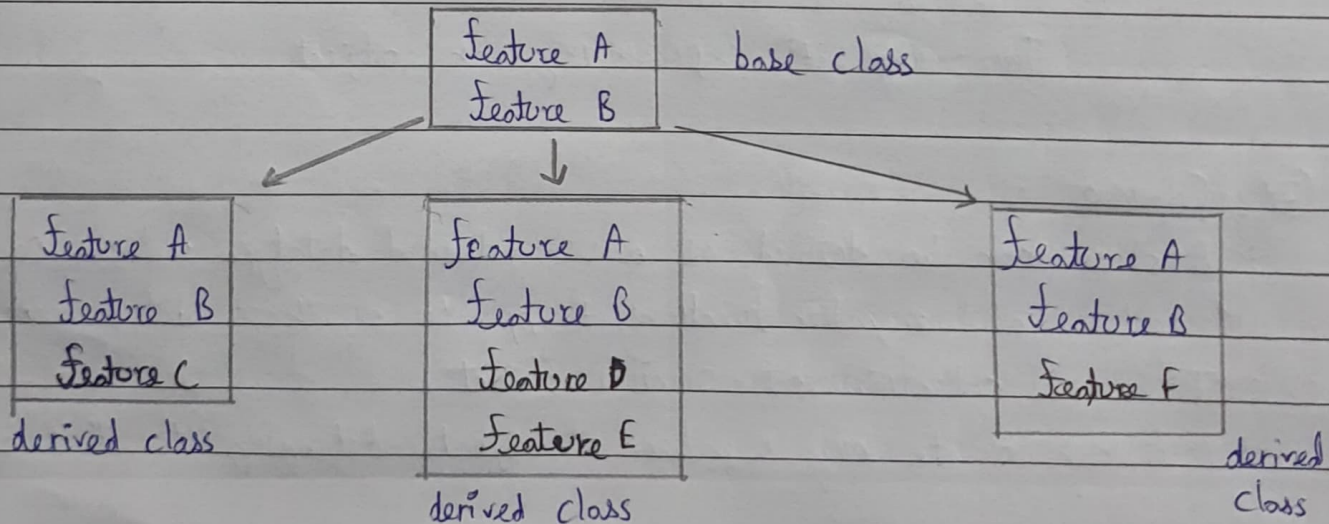
(3) Data Abstraction and Encapsulations

* The wrapping up of data and functions into a single unit (called class) is known as Encapsulation.
* Data encapsulation ensures that the data is only accessible to those functions which are wrapped in the class and not to the outside world

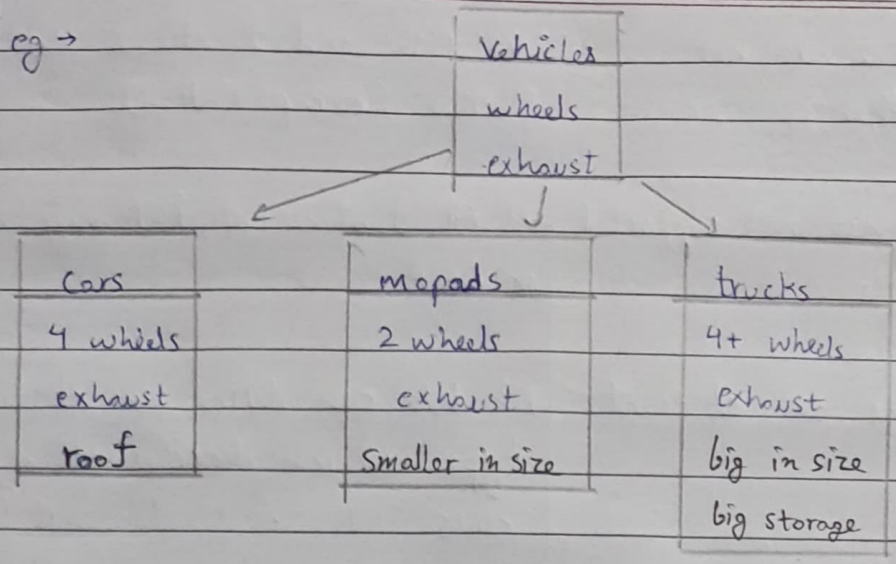
* Abstraction refers to the act of representing the essential features without including the background details or explanations.
eg → for a class person we may take the person's name, date of birth, and age without including details such as what they wear or where they live etc.

(4) Inheritance

* The mechanism of deriving a new class from a base class is called inheritance.
* The concept of inheritance provides the idea of reusability i.e adding new features to an ~~class~~ existing class without modifying it.



eg →



⑤ Polymorphism

Polymorphism is the concept of object oriented programming which refers to the ability of a method to behave differently in different instances.

eg → Consider the operation of addition,
for two numbers, the operation will generate the sum of the numbers but for two strings, the operation would produce a third string by concatenation.

Q11> Compare: Procedure Oriented Programming vs Object oriented programming

Procedure oriented Programming	Object oriented programming
i> It consists of writing list of instructions and organize them into groups called functions	i> It consists of building class and objects.
ii> Emphasis is on algorithm, data is not secure	ii> Emphasis is on data and not procedure, data is secure.
iii> This is good for scientific application development	iii> This is good for real-time software design.

iv> Data and functions are not combined together	iv> Data and functions are combined together (Encapsulation).
v> Features of OOP are not supported	v> All features of POP may be present in OOP
vi> We cannot define an abstract datatype.	vi> We can define an abstract data type which describes all objects of a class.
vii> Many Functions can access the same data.	vii> Object communicate with each other by message passing and data can only be accessed by the function wrapped in that class of the object.
viii> Top-down design concept is followed	viii> Bottom-up design concept is followed.
ix> Communication is done by parameters and return values.	ix> Communication is done by sending messages to the objects created.
x> data are exchanged between procedures under control of main program.	x> Objects exchange messages with each other.
xi> Easy to use, learn and understand	xi> Not that easy to use and understand thus not yet completely accepted by major vendors.
xii> It is time consuming, there is no code reusability, it is difficult to maintain	xii> It provides code reusability, reduced maintainance, real world modelling, less time consuming.