Class: D10A Seat No.____

Roll No. 01

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R C Marg, Chembur, Mumbai-400074

CERTIFICATE

Certified that Mr./~~Miss~~ **Aamir Z Ansari** of **D10A** has satisfactorily completed a course of the necessary experiments/assignments in **Computer Programming Paradigms Lab** under my supervision in the Institute of technology in the year 2020 - 2021

Principal Head of Department

Lab In-charge Subject Teacher

Vivekanand Education Society's Institute Of Technology
Department Of Information Technology
2020-2021

Name of the Course: CPPL
Year/Sem/Class: S.E.(INFT) Sem III (D10A)
Faculty In charge : Mrs. Vinita M. and Mrs. Dimple B.
Email: vinita.mishra@ves.ac.in, dimple.bohra@ves.ac.in

Sr. No.	Lab Experiments
1	Write a program based on Encapsulation, Initialization and Finalization.
2	Implementation of Inheritance
3	Program to illustrate Dynamic Binding
4	Haskell (Basics)
5	Functions in Haskell
6	Haskell (pattern Matching and Case expression)
7	Give the installation procedure of SWI prolog stepwise.
8	Prolog programs: a. write prolog code to check the relationship among individuals(son, daughter, mother, father, grandparent). Consider the facts of your interest and write the rules. Test the knowledge base with some goals. b. Write prolog code to find length of the list
A1	Write a short note on how to demonstrate thread management in Java. a. creation of thread b. Thread running
A2	Write a short note on demonstration of exception handling in Java.

Computer Programming Paradigm Lab
Lab Experiment No 01

Name: Aamir Ansari **Batch** A **Roll no.** 01

Aim: Write a C++ program to implement Triangle class which has the following members - three sides, four constructors (one with no parameter, one with single parameter (equilateral triangle), one with two parameters (isosceles triangle), one with three parameters (scalene triangle), a destructor, methods to read data and display data along with area of respective triangles.'

Theory:

1. Advantages of encapsulation

i) Data Hiding: Encapsulation protects an object from unwanted access by clients. Encapsulation allows access to a level without revealing the complex details below that level.

ii) Flexibility: With this, we can make the data as read-only or write-only as we require it to be. It also improves the maintainability and flexibility of code

iii) Reusability: It allows the user to use the existing code again and again in an effective way

iv) Testing of the code: Ease of testing, so it is better for Unit testing

2. Differentiate between object and class.

Class	Object
A class is a template for creating objects in program.	The object is an instance of a class.
A class is a logical entity	Object is a physical entity
A class does not allocate memory space when it is created.	Object allocates memory space whenever they are created.
You can declare class only once.	You can create more than one object using a class.

3. **Constructors, and its types**

A constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class.

Types of Constructor:

1. Default constructor: It is the constructor which doesn't take any argument. It has no parameters.
2. Parameterized Constructors: It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object.

4. **When are constructors invoked? How are they different from functions?**

Each time an instance of a class is created the constructor method is called. Constructors is a special member function of class and it is used to initialize the objects of its class. It is treated as a special member function because its name is the same as the class name. These constructors get invoked whenever an object of its associated class is created. It is named as "constructor" because it constructs the value of data member of a class. Initial values can be passed as arguments to the constructor function when the object is declared.

A constructor is different from normal functions in following ways:

1. Constructor has same name as the class itself
2. Constructors don't have return type
3. A constructor is automatically called when an object is created.
4. If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

5. **Explain the concept of constructor overloading.**

As we know function overloading is one of the core feature of the object oriented languages. We can use the same name of the functions; whose parameter sets are different. Here we will see how to overload the constructors of C++ classes. The constructor overloading has few important concepts.

Overloaded constructors must have the same name and different number of arguments
The constructor is called based on the number and types of the arguments are passed.
We have to pass the argument while creating objects, otherwise the constructor cannot understand which constructor will be called.

Program:

```
//code
#include <iostream>
#include <cmath>

using namespace std;

class Triangle {
   public:
      //Class members
      float s1, s2, s3;

      //Non-Parameterised
      Triangle() {
         float area = 0;
         s1 = 0;
         s2 = 0;
         s3 = 0;
      }

      //One parameter, EQUILATERAL
      Triangle(float side) {
         float area;
         s1 = side;
         s2 = side;
         s3 = side;
      }

      //Two parameters, ISOSCELES
      Triangle(float side1, float side2) {
         s1 = side1;
         s2 = side1;
         s3 = side2;
      }

      //Three parameters, SCALENE
      Triangle(float side1, float side2, float side3) {
         s1 = side1;
         s2 = side2;
         s3 = side3;
      }

      //Display side and area
      void area() {
         float a, s;
```

```
        s = 0.5*(s1+s2+s3);
        a = sqrt((s*(s-s1)*(s-s2)*(s-s3)));
        cout << "Area of triangle with sides  " << s1 << ", " << s2 << ", " << s3 << " is :   "
<< a << endl;
    }

    //destructor
    ~Triangle(){};

};

int main(){

   int n=0;
   float tempSides[3];
   cout << "Enter number of known sides :  " << endl;
   cout << "*0 Zero " << endl;
   cout << "*1 One Side (EQUILATERAL)" << endl;
   cout << "*2 Two Sides (ISOSCELES)" << endl;
   cout << "*3 Three Sides (SCALENE)" << endl;
   cin >> n;

   //input value of sides
   for (int i=0 ; i<n ; i++) {
      cout << "Enter side number " << i+1 << " :  ";
      cin >> tempSides[i];
   }

   //Send to different constructors
   switch(n) {
      case 0: {   //zero
         Triangle myObj;
         myObj.area();
         break;
      }
      case 1: {   //EQUILATERAL
         Triangle myObj(tempSides[0]);
         myObj.area();
         break;
      }
      case 2: {   //ISOSCELES
         Triangle myObj(tempSides[0], tempSides[1]);
         myObj.area();
         break;
      }
      case 3: {   //SCALENE
         Triangle myObj(tempSides[0], tempSides[1], tempSides[2]);
         myObj.area();
         break;
      }
      default:
```

```
            cout << "Invalid Input" << endl;
        }

        return 0;
    }
```

//output

```
Enter number of known sides :
*0 Zero
*1 One Side (EQUILATERAL)
*2 Two Sides (ISOSCELES)
*3 Three Sides (SCALENE)
0
Area of triangle with sides  0, 0, 0 is :   0



Enter number of known sides :
*0 Zero
*1 One Side (EQUILATERAL)
*2 Two Sides (ISOSCELES)
*3 Three Sides (SCALENE)
1
Enter side number 1 :  12
Area of triangle with sides  12, 12, 12 is :   62.3538


 Enter number of known sides :
 *0 Zero
 *1 One Side (EQUILATERAL)
 *2 Two Sides (ISOSCELES)
 *3 Three Sides (SCALENE)
 2
 Enter side number 1 :  10
 Enter side number 2 :  5
 Area of triangle with sides  10, 10, 5 is :   24.2061

Enter number of known sides :
*0 Zero
*1 One Side (EQUILATERAL)
*2 Two Sides (ISOSCELES)
*3 Three Sides (SCALENE)
3
Enter side number 1 :  4
Enter side number 2 :  7
Enter side number 3 :  6
Area of triangle with sides  4, 7, 6 is :   11.9765

Process returned 0 (0x0)   execution time : 15.118 s
Press any key to continue.
```

PCPF Lab
Lab Assignment number 02

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Implementation of Inheritance

Problem Statement:

Write the code to implement the concept of inheritance for Vehicles. You are required to implement inheritance between classes. There would be 3 classes - one superclass and two sub classes. Vehicle is the super class, whereas Bus and Truck are subclasses of Vehicle class.
Detailed description of Vehicle (Super class):
The class Vehicle must have following attributes (private):
1. Vehicle model
2. Registration number
3. Vehicle speed (km/hour)
4. Fuel capacity (liters)
5. Mileage (kilometers/liter)
The Vehicle class must have following functions:
1. Parameterized constructor that will initialize all the data members with the given values.
2. Getters and Setters for each data member that will get and set the values of data members of class.
3. A function fuelNeeded() that will take distance (in kilometer) as an argument. It will calculate the amount of fuel needed for the given distance and will return the value of fuel needed for given distance. You can use the attributes ‘ Mileage’ defined within the above Vehicle class to determine the fuel needed for the given distance. You are required to implement this functionality by yourself.
4. A function distanceCovered() that will take time (in hours) as an argument. It will calculate the distance for the given time and speed and returns the value of distance. The formula to calculate speed is given as speed = distance/time . You can use this formula to calculate the distance.
5. A display() function to display all details of the Vehicle.
Detailed description of Truck (Sub class):
The class Truck must have following attribute (private):
Cargo weight limit (Kilo grams)
The above class must have following functions:
1. Parameterized constructor that will initialize all data members with the given values.
2. Getters and setters for each data member that will get and set the values of data members of class.
3. It must override the display() function of Vehicle and display the details of the Truck object.
Detailed description of Bus (Sub class):
The class Bus must have following attribute (private):
No of passengers
The above class must have following functions:

1. Parameterized constructor that will initialize all the data members with given values.
2. Getters and setters that will get and set the value of each data member of class.
3. It must override the display() function of Vehicle and display the details of the Truck object.

In main function, perform the following:

● Create an instance of class Truck and initialize all the data members with proper values.
● Create an instance of class Bus and initialize all the data members with proper values.
● Now, call fuelNeeded () , distanceCovered () and display() functions using objects of these classes.

Theory:

Inheritance and its Advantages:

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important feature of Object Oriented Programming.

Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

Advantages of inheritance:

1. It allows the code to be reused as many times as needed
2. The base class once defined and once it is compiled, it need not be reworked
3. Saves time and effort as the main code need not be written again
4. Saves time in program development

Discuss with examples, the implications of deriving a class from an existing class by the private, public and protected access specifiers

Public mode: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.

Protected mode: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.

Private mode: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.

When Constructors and Destructors are executed, in base and consecutive derive classes

Constructor of base class is executed before that of derived class. Inn case of multiple inheritance, the order in which they are inherited is the order is the order of constructor calling
Destructor works the other way around

Difference: Overriding and Overloading

Overloading	Overriding
Method overloading is used to increase the readability of the program.	Method overriding is used to provide the specific implementation of the method that is already provided by its super class.
Method overloading is performed within class.	Method overriding occurs in two classes that have IS-A (inheritance) relationship.
In case of method overloading, parameter must be different.	In case of method overriding, parameter must be same.
Method overloading is the example of compile time polymorphism.	Method overriding is the example of run time polymorphism.

Program:

// code

```
#include <iostream>

using namespace std;

//Super class
class Vehicle {
    private:
        string vehicleModel;
        int regNum;
        float speed, fuel, mileage;

    public:
        //Initialization
        Vehicle(string vehicleModel, int regNum, float speed, float fuel, float mileage) {
            this->vehicleModel = vehicleModel;
            this->regNum = regNum;
            this->speed = speed;
            this->fuel = fuel;
            this->mileage = mileage;
        }

        //Model getter setter
        string getVehicleModel () {
            return this->vehicleModel;
        }
        void setVehicleModel (string vehicleModel) {
            this->vehicleModel = vehicleModel;
        }

        //regNum getter setter
        int getRegNum () {
            return this->regNum;
        }
        void setRegNum(int regNum) {
            this->regNum = regNum;
        }

        //speed getter setter
        float getSpeed() {
            return this->speed;
        }
        void setSpeed (float speed) {
            this->speed = speed;
        }

        //fuel getter setter
        float getFuel() {
            return this->fuel;
        }
        void setFuel(float fuel) {
            this->fuel = fuel;
        }
```

```
        //mileage getter setter
        float getMileage() {
           return this->mileage;
        }
        void setMileage(float mileage) {
           this->mileage = mileage;
        }

        //fuels needed in liter
        float fuelNeeded (float km) {
           return (this->mileage / km);
        }

        //Distance covered
        float distanceCovered (float hour) {
           return (this->speed * hour);
        }

        //Display information of vehicle
        virtual void display() {
           cout << "Vehicle Model :  " << this->vehicleModel << endl;
           cout << "Registration number :  " << this->regNum << endl;
           cout << "Vehicle Speed :  " << this->speed << " km/h" << endl;
           cout << "Vehicle Fuel :  " << this->fuel << " liters" << endl;
           cout << "Vehicle Mileage :  " << this->mileage << "km/l" << endl;
        }

};

//Sub class
class Truck : public Vehicle {
   private:
        float cargoWeightLimit;

   public:
        //Initializing constructor
        Truck (string vehicleModel, int regNum, float speed, float fuel, float mileage, float
cargoWeightLimit)
        :Vehicle(vehicleModel, regNum, speed, fuel, mileage) {
           this->cargoWeightLimit = cargoWeightLimit;
        }

        //cargoWeightLimit getter setter
        float getCargoWeightLimit() {
           return this->cargoWeightLimit;
        }
        void setCargoWeightLimit() {
           this->cargoWeightLimit = cargoWeightLimit;
        }

        //Display info of bus
```

```
        void display() override {
            cout << "Vehicle Model :  " << getVehicleModel() << endl;
            cout << "Registration number :  " << getRegNum() << endl;
            cout << "Vehicle Speed :  " << getSpeed() << " km/h" << endl;
            cout << "Vehicle Fuel :  " << getFuel() << " liters" << endl;
            cout << "Vehicle Mileage :  " << getMileage()  << " km/l" << endl;
            cout << "Cargo Weight Limit :  " << this->cargoWeightLimit << endl;
        }
};

//Sub class
class Bus : public Vehicle {
    private:
        int numPassenger;

    public:
        //Initializing constructor
        Bus (string vehicleModel, int regNum, float speed, float fuel, float mileage, int numPassenger)
        :Vehicle(vehicleModel, regNum, speed, fuel, mileage) {
            this->numPassenger = numPassenger;
        }

        //numPassenger getter setter
        int getNumPassenger() {
            return this->numPassenger;
        }
        void setNumPassenger() {
            this->numPassenger = numPassenger;
        }

        //Display info of truck
        void display() override {
            cout << "Vehicle Model :  " << getVehicleModel() << endl;
            cout << "Registration number :  " << getRegNum() << endl;
            cout << "Vehicle Speed :  " << getSpeed() << " km/h" << endl;
            cout << "Vehicle Fuel :  " << getFuel() << " liters" << endl;
            cout << "Vehicle Mileage :  " << getMileage()  << " km/l" << endl;
            cout << "Number of Passenger :  " << this-> numPassenger << endl;
        }

};

int main() {

    //created object; initialized with constructor
    Truck truckObj("Belaz", 2, 60, 50, 15, 80);

    //created object; initialized with constructor
    Bus busObj("Volvo", 1, 25, 50, 10, 100);

    //Display truck
    cout << "********** B U S **********" << endl;
```

```
    busObj.display();
    cout << "Fuel needed :  " << busObj.fuelNeeded(10) << endl;
    cout << "Distance covered :  " << busObj.distanceCovered(2) << endl;

    cout << endl;
    //Display bus
    cout << "********** T R U C K **********" << endl;
    truckObj.display();
    cout << "Fuel needed :  " << truckObj.fuelNeeded(15) << endl;
    cout << "Distance covered :  " << truckObj.distanceCovered(2) << endl;
}

// output
```

```
********** B U S **********
Vehicle Model :  Volvo
Registration number :  1
Vehicle Speed :  25 km/h
Vehicle Fuel :  50 liters
Vehicle Mileage :  10 km/l
Number of Passenger :  100
Fuel needed :  1
Distance covered :  50

********** T R U C K **********
Vehicle Model :  Belaz
Registration number :  2
Vehicle Speed :  60 km/h
Vehicle Fuel :  50 liters
Vehicle Mileage :  15 km/l
Cargo Weight Limit :  80
Fuel needed :  1
Distance covered :  120
```

PCPF Lab
Lab Assignment number 03

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Implementation of Dynamic Binding

Problem Statement:

Write a program to design a class representing the information regarding digital libraries having books and tapes. Book and Tape class are derived from a common class called Media. (Think over data members of Media)
Class Book Data Members - title, price, pages
Member functions -Parameterized constructors
-Display book details
Class Tape Data Members - title, price, time
Member functions -Parameterized constructors
- Display tape details
The program should demonstrate Runtime Polymorphism. (Hint: Use virtual function)

Theory:

Dynamic Binding

C++ provides facility to specify that the compiler should match function calls with the correct definition at the run time; this is called dynamic binding or late binding or run-time binding. Dynamic binding is achieved using virtual functions. Base class pointer points to derived class object. And a function is declared virtual in base class, then the matching function is identified at run-time using virtual table entry.

Runtime Polymorphism:

Runtime polymorphism: This type of polymorphism is achieved by Function Overriding. Function overriding on the other hand occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.
// C++ program for function overriding

Virtual Functions:

```
#include <bits/stdc++.h>
using namespace std;
class base  {
public:
   virtual void print ()
   { cout<< "print base class" <<endl; }
   void show ()
   { cout<< "show base class" <<endl; }
};
class derived : public base  {
public:
    void print ()
   { cout<< "print derived class" <<endl; }
   void show ()
   { cout<< "show derived class" <<endl; }
};
int main()  {
   base *bptr;
   derived d;
   bptr = &d;

   //virtual function, binded at runtime (Runtime polymorphism)
   bptr->print();

   // Non-virtual function, binded at compile time
   bptr->show();

   return 0;
}
// output
print derived class
show base class
```

Pure Virtual Functions:

Sometimes implementation of all function cannot be provided in a base class because we don't know the implementation. Such a class is called abstract class. For example, let Shape be a base class. We cannot provide implementation of function draw() in Shape, but we know every derived class must have implementation of draw(). Similarly an Animal class doesn't have implementation of move() (assuming that all animals move), but all animals must know how to move. We cannot create objects of abstract classes.
A pure virtual function (or abstract function) in C++ is a virtual function for which we don't have implementation, we only declare it. A pure virtual function is declared by assigning 0 in declaration. See the following example.

Program:

```
// code


#include <iostream>
#include <cstring>

using namespace std;

// parent class Media
class Media {
   public:
      // instance variables
      int id, yearOfPublish;

      // constructor
      Media(int id, int yearOfPublish) {
         this->id = id;
         this->yearOfPublish = yearOfPublish;
      }

      //  virtual method which is overloaded
      virtual void display() {}

};
// sub-class Book
class Book : public Media {
   public:
      // instance variables
      string title;
      float price;
      int pages;
      // constructor
      Book(int id, int yearOfPublish ,string title, float price, int pages)
      :Media(id, yearOfPublish) {
         this->title = title;
         this->price = price;
         this->pages = pages;
      }
      // method to display
      virtual void display() {
         cout << "Book's ID :  " << this->id << endl;
         cout << "Book's Year of publish :  " << this->yearOfPublish << endl;
         cout << "Book's title' :  " << this->title << endl;
         cout << "Book's price :  " << this->price << " rs" << endl;
         cout << "Number of pages in book :  " << this->pages << " pages" << endl;

      }
};

// sub-class Tape
class Tape : public Media {
   public:
```

```
        // instance variables
        string title;
        float price;
        float time;
        // constructor
        Tape (int id, int yearOfPublish, string title, float price, float time)
        :Media(id, yearOfPublish) {
            this->title = title;
            this->price = price;
            this->time  = time;
        }
        // method to display
        virtual void display() {
            cout << "Tape's ID :  " << this->id << endl;
            cout << "Tape's Year of publish :  " << this->yearOfPublish << endl;
            cout << "Tape's title' :  " << this->title << endl;
            cout << "Tape's price  :  " << this->price << " rs" << endl;
            cout << "Tape's length :  " << this->time << " hours" << endl;

        }
};

int main() {
    // object of Book
    Media *myBook = new Book(1001, 2077, "To sleep in sea of stars", 399.99, 425);
    myBook->display();

    cout << "*****************************************" << endl;

    // object of Tape
    Media *myTape = new Tape(2002, 2001, "2001 Space odessey", 449.99, 2.5);
    myTape->display();
}


// output
Book's ID :  1001
Book's Year of publish :  2077
Book's title' :  To sleep in sea of stars
Book's price :  399.99 rs
Number of pages in book :  425 pages
*****************************************
Tape's ID :  2002
Tape's Year of publish :  2001
Tape's title' :  2001 Space odessey
Tape's price  :  449.99 rs
Tape's length :  2.5 hours
```

PCPF Lab
Lab Assignment number 04

Name: Aamir Ansari **Batch:** A **Roll** no. 01

Aim: To execute the following on the prelude command prompt

Problem Statement:

1. Converts temperatures in C to F
2. Use map to convert a string into a list of Booleans, each element in the new list representing whether or not the original element was a lower-case character, that is, it should take the string aBCde and return [True,False,False,True,True].
3. Find factorial of number
4. Display square of numbers given in list

Solution:

Converts temperatures in C to F

```
convert :: Float->Float
convert n = (n*(9/5.0))+32
```

```
Prelude> convert n = (n*(9/5.0))+32
Prelude> convert 30
86.0
Prelude> convert 58
136.4
Prelude>
```

Use map to convert a string into a list of Booleans, each element in the new list representing whether or not the original element was a lower-case character, that is, it should take the string aBCde and return [True,False,False,True,True].

```
import Data.Char
convert x  = map(\x -> isLower x) x
```

```
Prelude> import Data.Char
Prelude Data.Char> convert x  = map(\x -> isLower x) x
Prelude Data.Char> convert "aAbB"
[True,False,True,False]
Prelude Data.Char> convert "AAbb"
[False,False,True,True]
Prelude Data.Char>
```

Find factorial of number

```
fac :: (Integral a) => a -> a
fac n = product [1..n]
```

```
Prelude> fac n = product [1..n]
Prelude> fac 5
120
Prelude> fac 4
24
Prelude>
```

Display square of numbers given in list

square x = map(^2) x

```
Prelude> square x = map(^2) x
Prelude> square [0,2,3]
[0,4,9]
Prelude> square [5, 8, 2]
[25,64,4]
Prelude>
```

PCPF Lab
Lab Assignment number 05

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Write Haskell program for the following

Problem Statement:

1. Find whether the read number is even or odd. Display "Even" is the no read in is even or else "Odd".
2. Generate Fibonacci series until a given number using recursive.
3. Find the sum of list of odd numbers in a list. Initialize the list. Report error if list is empty.

Solution:

Find whether the read number is even or odd. Display "Even" is the no read in is even or else "Odd"

evenOdd n = if mod n 2 == 0 then print "EVEN" else print "ODD"

```
Prelude> evenOdd n = if mod n 2 == 0 then print "EVEN" else print "ODD"
Prelude>
Prelude> evenOdd 5
"ODD"
Prelude> evenOdd 12
"EVEN"
Prelude> evenOdd 0
"EVEN"
```

Generate Fibonacci series until a given number using recursive.

```
fibonacci a b = a : fibonacci b (a+b)

main = do
 putStrLn "Enter number of elements needed"
 input<-getLine
 let n = (read input :: Int)
 putStrLn "Fibonacci series :  "
 print (take(n) (fibonacci 0 1))
```

```
Enter number of elements needed
5
Fibonacci series :
[0,1,1,2,3]
```

Find the sum of list of odd numbers in a list. Initialize the list. Report error if list is empty

```
sumOdd n | listSum n == 0 = print "EMPTY LIST"
         | otherwise = print (listSum n)

listSum :: [Int]->Int
listSum n | n == [] = 0
          | odd(head n) = (head n) + listSum(tail n)
          | otherwise = 0 + listSum(tail n)
```

PCPF Lab
Lab Assignment number 06

Name: Aamir Ansari **Batch:** A **Roll** no. 01

Aim: To study and implement pattern Matching and Case in haskell.

Theory: Pattern matching is a mechanism for checking a value against a pattern. A successful match can also deconstruct a value into its constituent parts. It is a more powerful version of the switch statement in Java and it can likewise be used in place of a series of if/else statements.

For example, if [1, 2] is matched against [0, bot], then 1 fails to match 0, so the result is a failed match. (Recall that bot, defined earlier, is a variable bound to _|_.) But if [1, 2] is matched against [bot, 0], then matching 1 against bot causes divergence (i.e. _|_).

The other twist to this set of rules is that top-level patterns may also have a Boolean guard, as in this definition of a function that forms an abstract version of a number's sign:

```
sign x | x > 0    = 1
       | x == 0  = 0
       | x < 0  = -1
```

Pattern matching at the function head is compiled, more or less, to case expressions in the function body. Further, your use of case expressions involves pattern matching.

```
-- Pattern Matching

not  True  =  False

not False = True


-- Case Expression

not x = case x of True -> False

                  False -> True
```

PROBLEM STATEMENT 1:

Write a Haskell program to consider a function safetail that behaves in the same way as tail, except that safetail maps the empty list to the empty list, whereas tail gives an error in this case. Define safetail using:

(a) a conditional expression;

(b) guarded equations;

(c) pattern matching.

CODE:

1. Conditional expression;

```
safetail xs = if null xs
then []
else tail xs
```

2. Guarded equations;

```
safetail xs | null xs = []

| otherwise = tail xs
```

3. Pattern matching;

```
safetail [] = []
safetail xs = tail xs
```

OUTPUT:

1.

```
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Ashwini Parwatikar\Desktop\H34
Prelude> :load a2.hs
[1 of 1] Compiling Main             ( a2.hs, interpreted )
Ok, one module loaded.
*Main> [1,2,3,4,5]
[1,2,3,4,5]
*Main> safetail[1,2,3,4,5]
[2,3,4,5]
*Main> null[]
True
*Main>
```

2.

```
Prelude> :load a2.hs
[1 of 1] Compiling Main             ( a2.hs, interpreted )
Ok, one module loaded.
*Main> safetail[1...5]

<interactive>:8:11: error:
    Variable not in scope: (...) :: Integer -> Integer -> a
*Main> safetail[1..5]
[2,3,4,5]
*Main>
```

3.

```
*Main> :load a2.hs
[1 of 1] Compiling Main             ( a2.hs, interpreted )
Ok, one module loaded.
*Main> safetail[11,12,13,14,15]
[12,13,14,15]
*Main>
```

PROBLEM STATEMENT 2:

Write a Haskell program to implement a simple calculator. (use case statement)

CODE:

```
solveEqn = do
        putStrLn "Enter number 1 "
        input1 <- getLine
        putStrLn "Enter number 2 "
        input2 <- getLine
        let a = (read input1 :: Int)
        let b = (read input2 :: Int)
        putStrLn "Enter the operator from(+,-,/,*)"
        oper <- getChar
        return (case oper of
                '+' -> (a + b)
```

```
            '-' -> (a - b)
             '/' -> (a `div` b)
            '*' -> (a * b))

main = do
eval <- solveEqn
print ("Answer is " ++show(eval))
```

OUTPUT:

```
Prelude> :load a1.hs
[1 of 1] Compiling Main             ( a1.hs, interpreted )

a1.hs:2:1: warning: [-Wtabs]
    Tab character found here, and in 18 further locations.
    Please use spaces instead.
  |
2 |         putStrLn "Enter number 1 "
  | ^^^^^^^^
Ok, one module loaded.
*Main> :main
Enter number 1
1
Enter number 2
3
Enter the operator from(+,-,/,*)
+
"Answer is 4"
```

PCPF Lab
Lab Assignment number 07

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: To install SWI Prolog

Theory:

Steps for installation of SWI-Prolog (Version 8.0.2-1) on Windows:

1. Using a web-browser navigate to SWI-Prolog home page: https://www.swi-prolog.org/
2. Mouse over Download and Click SWI-Prolog
   This will take you download page: https://www.swi-prolog.org/Download.html
3. Download the file named swipl-8.0.2-1.x64exe, under Binaries
4. In Windows double-click the file to install Prolog (or in a command shell run > swipl8.0.2-1.x64.exe).
5. Then, allow SWI-Prolog to install programs.
6. Select that SWI-Prolog should be included into the PATH variable for all users
7. Then to check your installation, as follows: -
   1. Call SWI-Prolog in a terminal shell, by typing in …
      > swipl
      which should open the interpreter, i.e., you should see ?-
   2. When you are in the SWI-Prolog interpreter, i.e. when you see ?-

PCPF Lab
Lab Assignment number 08

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: To execute programs in Prolog

Problem Statement 1:

Write prolog code to check the relationship among individuals (son, daughter, mother, father, grandparent). Consider the facts of your interest and write the rules. Test the knowledge base with some goals.

Program:

Knowledge Base (facts.pl):

```
% males and females
male(abraham).
male(clancy).
male(herb).
male(homer).
male(bart).
female(mona).
female(lisa).
female(jackie).
female(maggie).
female(marge).
female(patty).
female(selma).
female(ling).

% maritals
married(abraham, mona).
married(mona, abraham).
married(clancy, jackie).
married(jackie, clancy).
married(homer, marge).
married(marge, homer).

% parents
parent(abraham, herb).
parent(mona, herb).
parent(abraham, homer).
parent(mona, homer).
parent(clancy, marge).
parent(jackie, marge).
parent(clancy, patty).
parent(jackie, patty).
parent(clancy, selma).
parent(jackie, selma).
```

```
parent(homer, bart).
parent(marge, bart).
parent(homer, lisa).
parent(marge, lisa).
parent(homer, maggie).
parent(marge, maggie).
parent(selma, ling).

/* rules */
son(Child, Parent) :- male(Child), parent(Parent,Child).
daughter(Child, Parent) :- female(Child), parent(Parent,Child).
father(Parent, Child) :- male(Parent), parent(Parent,Child).
mother(Parent, Child) :- female(Parent), parent(Parent, Child).
brother(Sibling, Bro) :- male(Bro), father(Father,Sibling), father(Father, Bro), Bro \= Sibling,
mother(Mother, Sibling), mother(Mother, Bro).
sister(Sibling, Sis) :- female(Sis), father(Father, Sibling), father(Father, Sis), Sis \= Sibling,
mother(Mother, Sibling), mother(Mother, Sis).
grandmother(Grandmother, Grandchild) :- female(Grandmother), parent(Parent, Grandchild),
parent(Grandmother, Parent).
grandfather(Grandfather, Grandchild) :- male(Grandfather), parent(Parent, Grandchild),
parent(Grandfather, Parent).
```

Output:

```
?- son(bart,homer).
true .

?- son(lisa,homer).
false.

?- daughter(lisa,homer).
true .

?- daughter(lisa,marge).
true.

?- daughter(bart,marge).
false.
```

?- father(homer,maggie).
true .

?- mother(marge,maggie).
true.

?- sister(marge,selma).
true .

?- sister(patty,marge).
true .

?- grandmother(mona,bart).
true .

?- grandfather(abraham,lisa).
true .

?- brother(lisa,bart).
true .

?- sister(bart,maggie).
true .

Problem Statement 2:

Write Prolog code to find length of the list.

Program:

Knowledge Base (facts.pl):

```
%length of a list
len([],0).
len([_|T],N) :- len(T,X), N is X+1.
```

Output:

Queries:

?- len([1,2,3,4],Length).
Length = 4.

?- len([1,2,3,4,5,6,7,8],Length).
Length = 8.

PCPF Lab
Assignment number 01

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Java is a multi-threaded programming language which means we can develop multi-threaded programs using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

Handling threads: Each thread created/executed by a Java program is represented in the Java language through an instance of the class "Thread". A thread executes the code that has received on instantiation time through an instance of the class "Runnable". A thread starts executing itself after the invocation of its method start().

Creation of thread: A thread goes through various stages in its life cycle. For example, a thread is born, starts, runs, and then dies. There are two methods of creating a thread :

1. Create a thread by implementing runnable interface :

Step 1: Implement a run() method provided by a Runnable interface. This method provides an entry point for the thread.

public void run()

Step 2: Instantiate a Thread object using the following constructor

Thread(Runnable threadobj, String threadname);

Where, threadObj is an instance of a class that implements the Runnable in terface and threadName is the name given to the new thread.

Step 3: Once a Thread object is created, you can start it by calling start() method, which executes a call to run( ) method.

void start();

2. Create a thread by extending a thread class:

Step 1: Override run( ) method available in Thread class. This method provides an entry point for the thread.

public void run()

Step 2: Once Thread object is created, you can start it by calling start() method, which executes a call to run( ) method.

void start();

Thread Running:

The run() method of thread class is called if the thread was constructed using a separate Runnable object otherwise this method does nothing and returns. When the run() method calls, the code specified in the run() method is executed. You can call the run() method multiple times. It does not return any value.

Two ways to call run method :

1. **Call the run() method using the start() method:**

```
RunExp1 r1= new RunExp1();
Thread t1 = new Thread(r1);
t1.start();
```

2. **Call the run() method using the run() method itself:**

```
RunExp2 t1= new RunExp2();
t1.run();
```

PCPF Lab
Assignment number 02

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Exception Handling

When an Exception occurs the normal flow of the program is disrupted and the program / application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled. The Exception Handling in Java is one of the powerful mechanisms to handle the runtime errors so that normal flow of the application can be maintained. It is a mechanism to handle runtime errors such as:

1. ClassNotFoundException
2. IOException
3. SQLException
4. RemoteException etc.

There are three types of exceptions:

1. Checked exception : The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

2. Unchecked exception : The classes which inherit RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

3. Error : Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

There are five keywords which are used in handling exceptions in Java :

1. try : Java try block is used to enclose the code that might throw an exception. It must be used within the method. If an exception occurs at the particular statement of try block, the rest of the block code will not execute.

2. catch : Java catch block is used to handle the Exception by declaring the type of exception within the parameter. The declared exception must be the parent class exception or the generated exception type.

3. finally : Java finally block is a block that is used to execute important code such as closing connection, stream etc. Java finally block is always executed whether exception is handled or not.

4. throw : It is used to throw an exception.

5. throws : It is used to declare exceptions. It specifies that there may occur an exception in the method. It is always used with method signatures.

Example with try-catch block:

```
public class Test_exception_handling {
	public static void main(String args[]) {
		try {
			int data=100/0;
		}
		catch(ArithmeticException e) {
			System.out.println(e);
		}
		System.out.println("rest of the code...");
	}
}
```

Output:

```
java.lang.ArithmeticException: / by zero
rest of the code...
```

Example with try-catch-finally block :

```
public class Try_catch_finally {
	public static void main(String args[]) {
		try {
			int data=100/0;
		}
		catch(ArithmeticException e) {
			System.out.println(e);
		}
		finally {
			System.out.println("finally block is executed");
		}
		System.out.println("rest of the code...");
	}
}
```

Output :

```
java.lang.ArithmeticException: / by zero
finally block is executed
rest of the code...
```

Common scenarios of Java Exceptions:

1. ArithmeticException : If we divide any number by zero, there occurs an ArithmeticException.

2. NullPointerException : If we have a null value in any variable, performing any operation on the variable throws a NullPointerException.

3. NumberFormatException : The wrong formatting of any value may occur NumberFormatException. Suppose, a string variable that has characters, converting this variable into digit will occur NumberFormatException.

4. ArrayIndexOutOfBoundsException : If you are inserting any value in the wrong index, it would result in ArrayIndexOutOfBoundsException.

5. StringIndexOutOfBoundsException : If you are inserting any value in the wrong index of the string, it would result in StringIndexOutOfBoundsException.

Multi-catch block: A try block can be followed by one or more catch blocks. Each catch block must contain a different exception handler. At a time only one exception occurs and at a time only one catch block is executed.

Example :

```
public class MultipleCatchBlock {
	public static void main(String[] args) {
		try {
			int a[]=new int[5];
			a[5]=30/0;
		}
		catch(ArithmeticException e) {
			System.out.println("Arithmetic Exception occurs");
		}
		catch(ArrayIndexOutOfBoundsException e) {
			System.out.println("ArrayIndexOutOfBounds Exception occurs");
		}
		catch(Exception e) {
			System.out.println("Parent Exception occurs");
		}
		System.out.println("rest of the code");
	}
}
```

Output :

```
Arithmetic Exception occurs
rest of the code
```

Nested try block: The try block within a try block is known as nested try block in java. Sometimes a situation may arise where a part of a block may cause one error and the entire block itself may cause another error. In such cases, exception handlers have to be nested.

Example:

```
public class NestedTry {
	public static void main(String[] args) {
		int a[]=new int[5];
	try {
		try {
			a[5]=30/0;
		}
		catch(ArithmeticException e) {
			System.out.println("Arithmetic Exception occurs");
		}
		try {
			System.out.println(a[10]);
		}
		catch(ArrayIndexOutOfBoundsException e) {
			System.out.println("ArrayIndexOutOfBounds Exception occurs");
		}
	}
	catch(Exception e) {
		System.out.println("Parent Exception occurs");
	}
	System.out.println("rest of the code");
	}
}
```

Output:

```
Arithmetic Exception occurs
ArrayIndexOutOfBounds Exception occurs
rest of the code
```

User Defined exceptions: In java we can create our own exception class and throw that exception using throw keyword. These exceptions are known as user-defined or custom exceptions.