

### pcpf practical

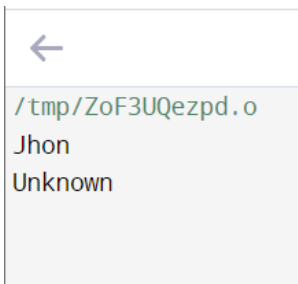
1) Write a program to print the names of students by creating a Student class. If no name is passed while creating an object of the Student class, then the name should be "Unknown", otherwise the name should be equal to the String value passed while creating the object of the Student class.

code -

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

class Student {
    string name;
public:
    Student(string s) {
        name = s;
    }
    Student() {
        name = "Unknown";
    }
    void print_name() {
        cout << name << endl;
    }
};

int main() {
    Student s1("Jhon");
    Student s2;
    s1.print_name();
    s2.print_name();
    return 0;
}
```



2) Create a class to print the area of a square and a rectangle. The class has two functions with the same name but different number of parameters. The function for printing the area of rectangle has two parameters which are its length and breadth respectively while the other function for printing the area of square has one parameter which is the side of the square.

code -

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

```
class Area {
public:
    void output(int l, int b) {
        cout<<"Area of Rectangle is = "<<l*b<<endl;
    }

    void output(int a) {
        cout<<"Area of Square is = "<<a*a<< endl;
    }
};
```

```
int main() {
    Area obj;
    obj.output(5,6);
    obj.output(5);
}
```

```
←  
/tmp/MNw1LDw1V8.o  
Area of Rectangle is = 30  
Area of Square is = 25
```

3) We want to calculate the total marks of each student of a class in Physics, Chemistry and Mathematics and the average marks of the class. The number of students in the class are entered by the user. Create a class named Marks with data members for roll number, name and marks. Create three other classes inheriting the Marks class, namely Physics, Chemistry and Mathematics, which are used to define marks in individual subject of each student.

code -

```
#include<iostream>  
using namespace std;  
class Physics;  
class Chemistry;  
class Mathematics;  
class Marks{  
    protected:  
        int rollno;  
        string name;  
        int marks[3];  
    public:  
        Marks(){  
        }  
        Marks(int roll, string n){  
            rollno = roll;  
            name = n;  
        }  
        void display(){  
            cout<<"Roll Number:  "<<rollno<<" \n ";  
            cout<<"Name:  "<<name<<" \n ";
```

```

        }
        friend int getTotalMarks(Marks &, Physics&, Chemistry&,
Mathematics&);
};
class Physics: public Marks{
    private:
        int mark;
    public:
        Physics(){
        }
        Physics(int mark){
            this->mark = mark;
        }
        friend int getPhysicsMarks(Physics &);

};
class Chemistry: public Marks{
    private:
        int mark;
    public:
        Chemistry(){
        }
        Chemistry(int mark){
            this->mark = mark;
        }
        friend int getChemistryMarks(Chemistry &);

};
class Mathematics: public Marks{
    private:
        int mark;
    public:
        Mathematics(){
        }
        Mathematics(int mark){

```

```

        this->mark = mark;
    }
    friend int getMathematicsMarks(Mathematics &);
};

//The function to return Physics marks
int getPhysicsMarks(Physics &p){
    return p.mark;
}

//The function to return Chemistry marks
int getChemistryMarks(Chemistry &c){
    return c.mark;
}

//The function to return Mathematics mark
int getMathematicsMarks(Mathematics &m){
    return m.mark;
}

int getTotalMarks(Marks&ma,Physics&p, Chemistry&c,
Mathematics&m){

    int first = getPhysicsMarks(p);
    int second = getChemistryMarks(c);
    int third = getMathematicsMarks(m);
    ma.marks[0] = first;
    ma.marks[1] = second;
    ma.marks[2] = third;
    int sum = 0.0;
    for(int i=0; i<3; i++){
        sum += ma.marks[i];
    }
    return sum;
}

int main(){

```

```

int n;
cout<<"Enter the total number of students in the class\n";
cin>>n;
double totalSum = 0.0;
int physics;
int chemistry;
int maths;
string name;
Marks marks[n];
double total [n];
for(int i=0; i<n; i++){
    cout<<"Student "<<(i+1)<<endl;
    cout<<"Enter the student name: "<<endl;
    cin>>name;
    Marks J(i+1,name);
    marks[i] = J;
    cout<<"Enter the student Physics marks: "<<endl;
    cin>>physics;
    cout<<"Enter the student Chemistry marks: "<<endl;
    cin>>chemistry;
    cout<<"Enter the student Mathematics marks: "<<endl;
    cin>>maths;
    Physics p1(physics);
    Chemistry c1(chemistry);
    Mathematics m1(maths);
    total[i] = getTotalMarks(marks[i],p1,c1,m1);
}
for(int i=0; i<n; i++){
    cout<<"Student: "<<(i+1)<<endl;
    marks[i].display();
    cout<<"Total marks: "<<total[i]<<endl;
}
double average = 0.0;
for(int i=0; i<n; i++){
    totalSum += total[i];
}

```

```
}  
cout<<"Total marks of the class is: "<<totalSum<<"\nThe average  
marks of the class is: "<<totalSum /n;  
}
```

output:

/tmp/brdPGTvlG5.o

Enter the total number of students in the class

3

Student 1

Enter the student name:

kunal

Enter the student Physics marks:

78

Enter the student Chemistry marks:

65

Enter the student Mathematics marks:

78

Student 2

Enter the student name:

sumeet

Enter the student Physics marks:

57

Enter the student Chemistry marks:

86

Enter the student Mathematics marks:

12

Student 3

Enter the student name:

sujal

Enter the student Physics marks:

78

Enter the student Chemistry marks:

35

Enter the student Mathematics marks:

45

Student: 1

Roll Number: 1

Name: kunal

Total marks: 221

Student: 2

Roll Number: 2

Name: sumeet

Total marks: 155

Student: 3

Roll Number: 3

Name: sujal

Total marks: 158

Total marks of the class is: 534

The average marks of the class is: 178

4) Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two functions to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize the length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class. Print the area and perimeter of a rectangle and a square.

code -

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

```
class Rectangle
{
    int length,breadth;
public:
    Rectangle(int l, int b)
    {
```



```

        length = l;
        breadth = b;
    }

    void print_area()
    {
        cout << length*breadth << endl;
    }

    void print_perimeter()
    {
        cout << 2*(length+breadth) << endl;
    }
};

class Square : public Rectangle
{
public:
    Square(int side) : Rectangle(side,side)
    {}
};

int main()
{
    Rectangle r(4,5);
    Square s(4);
    r.print_area();
    r.print_perimeter();
    s.print_area();
    s.print_perimeter();
    return 0;
}

```

```
←  
/tmp/7FkS1rQDgG.o  
20  
18  
16  
16
```

## 5. input:

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

//wap to print the sum,difference and product of 2 complex nos. Create a class Complex, with separate methods for each operation. Real and imaginary parts are entered by the user.Implement via operator overloading.

//operator overloading is used when redefining the meaning of the operators is needed.the main idea behind “Operator overloading” is to use c++ operators with class variables or class objects. Redefining the meaning of operators really does not change their original meaning; instead, they have been given additional meaning along with their existing ones.

```
class Complex{  
    int real, imag;  
  
    public:  
  
    Complex(int r = 0, int i = 0) {  
        real = r;  
        imag = i;  
    }  
  
    Complex operator + (Complex const &obj) {  
        Complex res;  
        res.real = real + obj.real;  
        res.imag = imag + obj.imag;  
        return res;  
    }  
  
    Complex operator - (Complex const &obj) {  
        Complex res;  
        res.real = real - obj.real;  
        res.imag = imag - obj.imag;  
        return res;  
    }  
}
```

```

Complex operator * (Complex const &obj) {
    Complex res;
    res.real = (real * obj.real) - (imag * obj.imag);
    res.imag = (imag * obj.real)+ (real * obj.imag);
    return res;
}

void print() {
    cout <<"Result : " <<real << " + i" << imag << "\n";
}
};

int main() {
    Complex c1(10, 5), c2(3, 2);

    Complex c3 = c1 + c2;
    c3.print();

    Complex c4 = c1 - c2;
    c4.print();

    Complex c5 = c1 * c2;
    c5.print();

    return 0;
}

```

## 5.output:

```

Output
/tmp/hpvQNjhTQX.o
Result : 13 + i7
Result : 7 + i3
Result : 20 + i35

```

6)Write a C++ program to demonstrate different access specifiers in C++

code -

**for public -**

```
#include <iostream>
```

```
using namespace std;
```

```
// define a class
class Sample {

    // public elements
public:
    int age;

    void displayAge() {
        cout << "Age = " << age << endl;
    }
};

int main() {


    // declare a class object
    Sample obj1;

    cout << "Enter your age: ";

    // store input in age of the obj1 object
    cin >> obj1.age;

    // call class function
    obj1.displayAge();

    return 0;
}
```

A screenshot of a terminal window with a dark background. It shows the output of the program: "Enter your age: 20" followed by "Age = 20" on the next line.

```
Enter your age: 20
Age = 20
```

**for private -**

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

// define a class
class Sample {

    // private elements
private:
    int age;

    // public elements
public:
    void displayAge(int a) {
        age = a;
        cout << "Age = " << age << endl;
    }
};

int main() {

    int ageInput;

    // declare an object
    Sample obj1;

    cout << "Enter your age: ";
    cin >> ageInput;

    // call function and pass ageInput as argument
    obj1.displayAge(ageInput);

    return 0;
}
```

```
Enter your age: 20  
Age = 20
```

### **for protected -**

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

// declare parent class
class Sample {
    // protected elements
protected:
    int age;
};

// declare child class
class SampleChild : public Sample {

public:
    void displayAge(int a) {
        age = a;
        cout << "Age = " << age << endl;
    }

};

int main() {
    int ageInput;

    // declare object of child class
    SampleChild child;

    cout << "Enter your age: ";
```

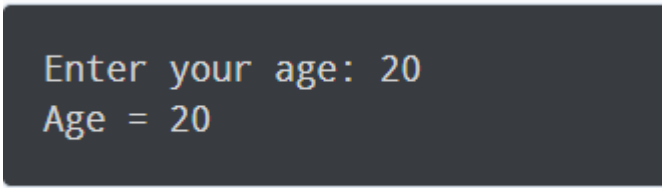
```

    cin >> ageInput;

    // call child class function
    // pass ageInput as argument
    child.displayAge(ageInput);

    return 0;
}

```



A screenshot of a terminal window with a dark background. It shows the prompt 'Enter your age: 20' and the output 'Age = 20'.

## 7. Input:

```

#include <iostream>
using namespace std;

//wap to accept details of a person incl age, throw an exception if age is lesser than 18.
otherwise display "You can vote."

int main() {
    string name;
    cout << "Enter your name : ";
    cin>> name;

    try{
        int age;
        cout << "Enter your age : ";
        cin>> age;
        if(age<18){
            throw (age);
        }
        else cout<<"You can vote."<< endl;
    }

    catch(...){
        cout<<"Underage, you cannot vote."<< endl;
    }

    return 0;
}

```

## 7. OUTPUT:

### Output

```
/tmp/8liw50Qj31.o
Enter your name : Tim
Enter your age : 13
Underage, you cannot vote.
|
```

### Output

```
/tmp/8liw50Qj31.o
Enter your name : Sam
Enter your age : 23
You can vote.
```

## HASKELL

1) Write a Haskell program to find the factorial of a number using pattern matching, guards and simple control statements  
code -

### pattern matching -

```
factorial :: Int -> Int
factorial 0 = 1
factorial n = n * factorial (n - 1)
```

```
main = print (factorial 5)
```

### Output:

```
GHCI, version 9.4.2: https://www.haskell.org/ghc/  :? for help
ghci> :l factorial.hs
[1 of 2] Compiling Main                ( factorial.hs, interpreted )
Ok, one module loaded.
ghci> main
120
```

### guards -

```
factorial :: Int -> Int
factorial x | x == 0 = 1
factorial x | x /= 0 = x * factorial (x - 1)
```

```
main = print (factorial 4)
```

### Output:

Same as above



## control statements-

```
factOn :: Int -> Int
factOn n = if n == 0
then 1
else n * factOn (n-1)
```

```
main = print( factOn 5)
```

### Output:

Same as above

2)Write a Haskell program to implement a user defined function to find the sum of list of numbers.

### code -

```
sumList :: [Int] -> Int
sumList [] = 0
sumList (x:xs) = x + sumList xs
main = print (sumList [1,2356])
```

### Output:

```
ghci> :l factorial.hs
[1 of 2] Compiling Main                ( factorial.hs, interpreted )
Ok, one module loaded.
ghci> main
2357
ghci> 
```

3)Write a Haskell program to implement a user defined function to find the number of elements in a list.

### code -

```
listLength :: [a] -> Integer
listLength [] = 0
listLength (_:xs) = 1 + listLength xs
main = print(listLength [1,2,3,4,5,6,7])
```

### Output:

```
ghci> :l factorial.hs
[1 of 2] Compiling Main                ( factorial.hs, interpreted )
Ok, one module loaded.
ghci> main
7
ghci> 
```

4) Write a Haskell program to implement a user defined function to find whether is number is Prime or Not.

**INPUT:**

```
factors n = [x | x <- [1..n], n `mod` x == 0]
```

```
prime n = factors n == [1,n]
```

```
main = do
```

```
putStrLn "hello world"
```

```
print (prime 3)
```

```
print (prime 4)
```

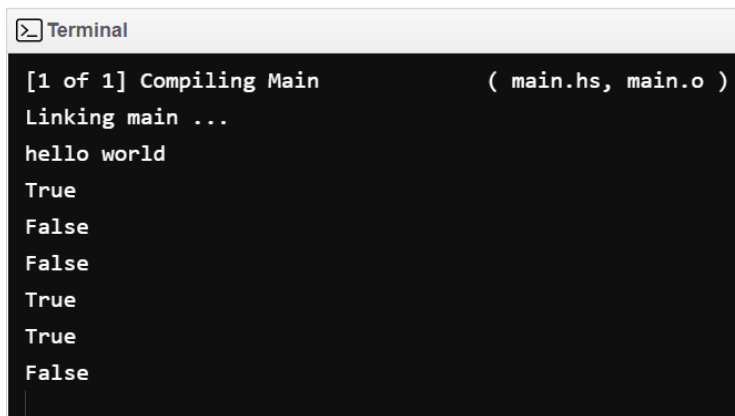
```
print (prime 1)
```

```
print (prime 2)
```

```
print (prime 11)
```

```
print (prime 15)
```

**OUTPUT:**

A terminal window titled "Terminal" with a dark background. It shows the compilation and execution of a Haskell program. The output is as follows:

```
[1 of 1] Compiling Main           ( main.hs, main.o )
Linking main ...
hello world
True
False
False
True
True
False
```

5)Write a Haskell program to implement a user defined function to find the largest between 3 numbers. Accept the 3 numbers as input from the user

code -

```
smallest :: Int -> Int -> Int -> Int
```

```
smallest a b c = min a (min b c)
```

```
main = print(smallest 100 450 78)
```

**Output:**

```
ghci> :l factorial.hs
[1 of 2] Compiling Main                ( factorial.hs, interpreted )
Ok, one module loaded.
ghci> main
78
ghci> 
```

6) Solve some basic problems using built in methods in haskell like (using all methods discussed in class)

-Print table of 9

**INPUT:**

```
multTable num = [(num * x) | x <- [1..10]]
```

```
main = do
```

```
putStrLn " Table of 9 :"
```

```
print (multTable 9)
```

**OUTPUT:**

```
Terminal

[1 of 1] Compiling Main                ( main.hs, main.o )
Linking main ...
Table of 9 :
[9,18,27,36,45,54,63,72,81,90]
```

-Display reverse of the string

**INPUT:**

```
reverseString :: [Char] -> [Char]
```

```
reverseString [] = []
```

```
reverseString (x:xs) = reverseString xs ++ [x]
```

```
main = print(reverseString['a', 'b', 'c'])
```

**Output:**

```
ghci> :l factorial.hs
[1 of 2] Compiling Main                ( factorial.hs, interpreted )
Ok, one module loaded.
ghci> main
"cba"
ghci> 
```

7) Solve some basic problems with lists in haskell like

### **-find the last element is the list**

```
main = do
  let x = [1,3,5,7,9,82,92,473]
  print(last x)
```

#### **Output:**

```
ghci> :l factorial.hs
[1 of 2] Compiling Main                ( factorial.hs, interpreted )
Ok, one module loaded.
ghci> main
473
ghci> 
```

### **comprehension list to generate a list**

Code-

```
square = [(x * x) | x <- [1..10]]
main = do
  putStrLn " List of squares : "
  print (square)
```

#### **output:**

```
Terminal

[1 of 1] Compiling Main                ( main.hs, main.o )
Linking main ...
List of squares :
[1,4,9,16,25,36,49,64,81,100]
```

## **PROLOG**

1)Write a Prolog program to find the summing elements of a list of numbers  
code -

```
list_sum([],0).
list_sum([Head|Tail], TotalSum):-
  list_sum(Tail, Sum1),
  TotalSum is Head+Sum1.
```

## OUTPUT:



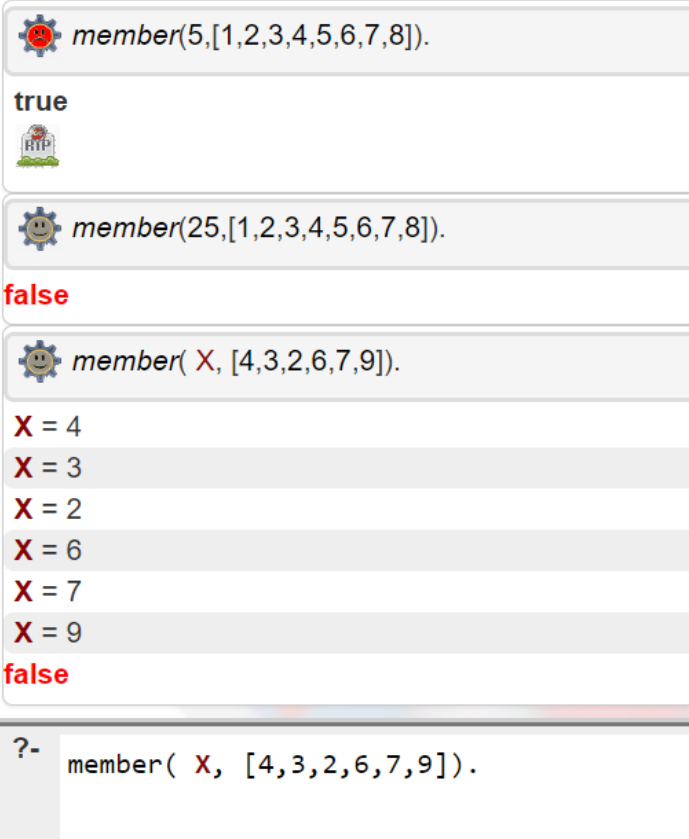
```
list_sum([2,4,6,8], TotalSum).  
TotalSum = 20  
list_sum([], TotalSum).  
TotalSum = 0  
?- list_sum([], TotalSum).
```

2)Write a Prolog program to find the list membership

Code-

```
member(X, [Y|T]) :- X = Y; member(X, T).
```

## OUTPUT:



```
member(5,[1,2,3,4,5,6,7,8]).  
true  
member(25,[1,2,3,4,5,6,7,8]).  
false  
member(X,[4,3,2,6,7,9]).  
X = 4  
X = 3  
X = 2  
X = 6  
X = 7  
X = 9  
false  
?- member(X,[4,3,2,6,7,9]).
```

3)Write a Prolog program to find the reversing a list

Code-

```
reverse([],Z,Z).
```

```
reverse([H|T],Z,Acc) :- reverse(T,Z,[H|Acc]).
```

## OUTPUT:

```
reverse([a,b,c],X,[]).
```

**X** = [c, b, a]

```
?- reverse([a,b,c],X,[]).
```

4)Write a Prolog program to find the length of a list  
code -

```
list_length([], 0).
list_length([_ | L], N) :-
    list_length(L, N1),
    N is N1 + 1.
```

**OUTPUT:**

```
list_length([],M), list_length([2,3,4,1],O), list_length([2,3,4,5,6,7,7,8],N).
```

**M** = 0,  
**N** = 8,  
**O** = 4

```
?- list_length([],M),
    list_length([2,3,4,1],O),
    list_length([2,3,4,5,6,7,7,8],N).
```

5)Write an effective prolog program to find the largest between 2 numbers  
code -

```
gt(X,Y,Z) :- Z is max(X,Y) .
```

**OUTPUT:**

```
gt(7,14,H), gt(18,13, W).
```

**H** = 14,  
**W** = 18

```
?- gt(7,14,H),
    gt(18,13, W).
```

```
gt(7,14,14).
```

true

```
gt(4,5,4).
```

false

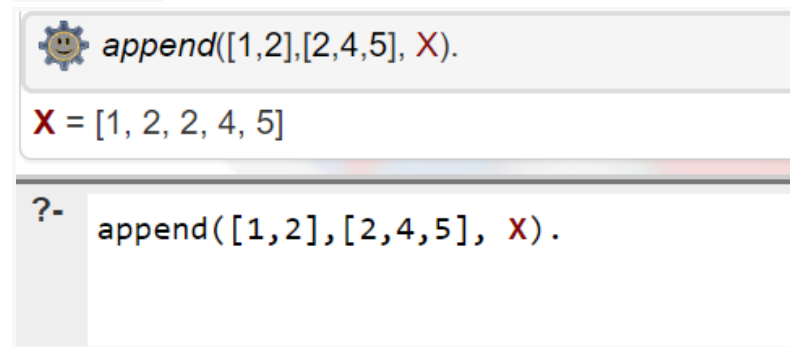
```
?- gt(4,5,4).
```

6)Write a Prolog program to append 2 lists.

code -

```
:- initialization(main).  
main :- write('append').  
append( [], X, X).  
append( [X | Y], Z, [X | W]) :- append( Y, Z, W).  
Input:  
append([1,2],[3,4,5], X).
```

**OUTPUT:**



```
append([1,2],[2,4,5], X).  
X = [1, 2, 2, 4, 5]  
?- append([1,2],[2,4,5], X).
```

7)Write rules to obtain new inference based on given knowledge base.

code -

%Facts

```
studies(charlie, csc135). % charlie studies csc135  
studies(olivia, csc135). % olivia studies csc135  
studies(jack, csc131). % jack studies csc131  
studies(arthur, csc134). % arthur studies csc134
```

```
teaches(kirke, csc135). % kirke teaches csc135  
teaches(collins, csc131). % collins teaches csc131  
teaches(collins, csc171). % collins teaches csc171  
teaches(juniper, csc134). % juniper teaches csc134
```

%Rules

```
professor(X, Y) :-  
teaches(X, C), studies(Y, C).
```

% X is a professor of Y if X teaches C and Y studies C.

**OUTPUT:**

 `professor(X,charlie), professor(collins,Y).`

`X = kirke,`

`Y = jack`

`false`

?- `professor(X,charlie),`  
`professor(collins,Y).`

% "false" appears coz y=jack doesn't study the second course that prof collins teaches.