# TASK 1:

**PERSON HEADER:**

#pragma once

#include<iostream>

using namespace std;

class person

{

public:

person();

person(int x);

};

**PERSON CPP:**

#include"Person.h"

person::person() {}

person::person(int x)

{

cout << "person::person(int x) called." << endl;

}

**FACULTY HEADER:**

#pragma once

#include"Person.h"

#include<iostream>

using namespace std;

class faculty : virtual public person

{

public:

faculty(int x);

};

**FACULTY CPP:**

#include"Faculty.h"

faculty::faculty(int x) :person(x)

{

cout << "faculty::faculty(int x) called" << endl;

}

**STUDENT HEADER:**

#pragma once

#include"Person.h"

#include<iostream>

using namespace std;

class student : virtual public person

{

public:

student(int x);

};

**STUDENT CPP:**

#include"Student.h"

student::student(int x) :person(x)

{

cout << "student::student(int )called" << endl;

}

**TA HEADER:**

#pragma once

#include"Faculty.h"

#include"Student.h"

#include<iostream>

using namespace std;

class TA : public faculty, public student

{

public:

TA(int x);

};

**TA CPP:**

#include"TA.h"

TA::TA(int x) :student(x), faculty(x), person(x)

{

cout << "TA::TA(int ) called" << endl;

}

**MAIN:**

#include<iostream>

#include"TA.h"

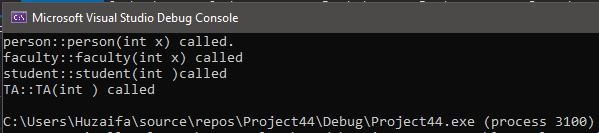
int main()

{

TA ta1(30);

}

# OUTPUT:



# TASK 2:

**ANIMAL HEADER:**

#pragma once

#include<iostream>

using namespace std;

class animals

{

string name;

public:

animals();

void sound();

animals(string name);

string getname();

}; **ANIMAL CPP:**

#include"animal.h"

animals::animals() {}

void animals::sound()

{

cout << "Animal sound " << endl;

}

animals :: animals(string name)

{

this->name = name;

}

string animals::getname()

{

return this->name;

} **CAT HEADER:**

#pragma once

#include<iostream>

using namespace std;

class cat : public animals

{

public:

cat();

void sound();

};

**CAT CPP:**

#include"cat.h"

#include"animal.h"

cat::cat() : animals("cat") {}

void cat::sound()

{

cout << "CAT Meow" << endl;

}

**DOG HEADER:**

#pragma once

#include<iostream>

using namespace std;

class dog : public animals

{

public:

dog();

void sound();

}; **DOG CPP:**

#include"dog.h"

#include"animal.h"

dog::dog() : animals("dog") {}

void dog::sound()

{

cout << "DOG Bark" << endl;

}

**DEER HEADER:**

#pragma once

#include<iostream>

using namespace std;

class deer : public animals

{

public:

deer();

void sound();

}; **DEER CPP:**

#include"Deer.h"

#include"animal.h"

deer::deer() {}

void deer::sound()

{

cout << "DEER Bleeh" << endl;

}

**TIGERFAMILY HEADER:**

#pragma once

#include<iostream>

using namespace std;

class TigerFamily : public animals

{

public:

TigerFamily();

void sound();

};

**TIGERFAMILY CPP:**

#include"TigerFamily.h"

#include"animal.h"

TigerFamily::TigerFamily() {}

void TigerFamily::sound()

{

cout << "TIGER Roar" << endl;

}

**TIGER HEADER:**

#pragma once

#include<iostream>

using namespace std;

class tiger : public TigerFamily

{

public:

tiger();

void sound();

}; **TIGER CPP:**

#include"Tiger.h"

#include"TigerFamily.h"

tiger::tiger() {}

void tiger::sound()

{

cout << "TIGER Roarr" << endl;

} **LION HEADER:**

#pragma once

#include<iostream>

using namespace std;

class lion : public TigerFamily

{

public:

lion();

void sound();

}; **LION CPP:**

#include"Lion.h"

#include"TigerFamily.h"

lion::lion() {}

void lion::sound()

{

cout << "LION Roarrr" << endl;

}

**SOURCE:**

#include<iostream>

using namespace std;

#include"animal.h"

#include"cat.h"

#include"dog.h"

#include"TigerFamily.h"

#include"Deer.h"

#include"Tiger.h"

#include"lion.h"

int main()

{

animals s("cat"), m;

m.sound();

cat a;

a.sound();

dog b;

b.sound();

TigerFamily t;

t.sound();

deer d;

d.sound();

tiger T;

T.sound();

lion lio;

lio.sound();

}

# OUTPUT:

# 

# 

# TASK 3:

**SHIP HEADER:**

#pragma once

#include<iostream>

using namespace std;

class ship

{

public:

string name;

string year;

ship();

void namesetter(string name);

void yearsetter(string year);

string namegetter();

string yeargetter();

virtual void print();

};

**SHIP CPP:**

#include"Ship.h"

ship::ship() {}

void ship::namesetter(string name)

{

this->name = name;

}

void ship::yearsetter(string year)

{

this->year = year;

}

string ship::namegetter()

{

return name;

}

string ship::yeargetter()

{

return year;

}

void ship::print()

{

cout << "Name of ship is:" << namegetter() << endl;

cout << "Year of ship is: " << yeargetter() << endl;

}

**CARGOSHIP HEADER:**

#pragma once

#include<iostream>

using namespace std;

class cargoShip :public ship

{

public:

int tonn;

cargoShip();

void tonnsetter(int tonn);

int tonngetter();

void print();

}; **CARGOSHIP CPP:**

#include"CargoShip.h"

#include"Ship.h"

cargoShip::cargoShip() {}

void cargoShip::tonnsetter(int tonn)

{

this->tonn = tonn;

}

int cargoShip::tonngetter()

{

return tonn;

}

void cargoShip::print()

{

ship::print();

cout << "Capacity of ship is: " << tonngetter() << endl;;

} **CRUISESHIP HEADER:**

#pragma once

#include<iostream>

using namespace std;

class cruiseShip : public ship

{

public:

int max;

cruiseShip();

void maxsetter(int max);

int maxgetter();

void print();

}; **CRUISESHIP CPP:**

#include"CruiseShip.h"

#include"Ship.h"

cruiseShip::cruiseShip() {}

void cruiseShip::maxsetter(int max)

{

this->max = max;

}

int cruiseShip::maxgetter()

{

return max;

}

void cruiseShip::print()

{

ship::print();

cout << "Maximum capacity is: " << maxgetter() << endl;;

} **SOURCE:**

#include<iostream>

#include"Ship.h"

#include"CargoShip.h"

#include"CruiseShip.h"

using namespace std;

int main()

{

ship\* obj[3];

obj[0] = new ship;

obj[0]->namesetter("SHIP");

obj[0]->yearsetter("2022");

obj[1] = new cruiseShip;

((cruiseShip\*)obj[1])->namesetter("CRUISE SHIP");

((cruiseShip\*)obj[1])->maxsetter(100);

((cruiseShip\*)obj[1])->yearsetter("2022");

obj[2] = new cargoShip;

((cargoShip\*)obj[2])->namesetter("CARGO SHIP");

((cargoShip\*)obj[2])->tonnsetter(500);

((cargoShip\*)obj[2])->yearsetter("2022");

for (int i = 0; i < 3; i++)

{

obj[i]->print();

cout << endl;

}

# } OUTPUT:

