Object Oriented Programming

FAST-ISB

Code Dry Runs

- This file contains code snippets for OOP concepts.
- Starting from pointers and Recursion.
- All important OOP concepts are focused including:
  - Constructors & Destructors
  - Copy Constructors, Shallow & Deep Copy
  - Static, Constant Member Functions
  - Operator Overloading
  - Object Typecasting
  - Association
  - Composition
  - Aggregation
  - Inheritance & its types
  - Polymorphism
  - Diamond Problem
- For better understanding of code snippets it is recommended to run the code snippets on debugger and visualize step by step execution.
- Python tutor Debugger (https://pythontutor.com) is highly recommended for step by step execution of codes and visualization.
- Made with ❤ by Aneeq Malik

Question – 1:

```
void mystery(int* ptr, int s)
{
	ptr = new int[s];
	for (int i = 0, j = s; i < s; ++i, j--)
		*(ptr + i) = j;
}
int main()
{
	int* ptr, s = 5;
	mystery(ptr, s);
	for (int i = 0; i < s; ++i)
		cout << ptr[i] << " ";
	delete[] ptr; ptr = NULL;
	return 0;
}
```

Question – 2:

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

char c[7][11] = { "OOP-Final", "OOP" ,"Exam" , "Students"," lazy" ,"2023" ,"programmer" };

char* add(char* ptr) {
	return ptr + 11;
}
char* sub(char* ptr) {
	return ptr - 11;
}
int main()
{
	char* mystery = c[4];
	cout<< mystery << endl;
	cout<< sub(mystery)[2] << endl;
	mystery = sub(mystery);
	cout<< mystery << endl;
	cout<< sub(mystery) + 1 << endl;
	cout<< add(add(mystery)) + 13 << endl;
	cout<< *add(add(mystery)) << endl;
	return 0;
}
```

Question – 3:

```
const char* c[] = { "OOP", "Exam" , "Oopsmid-1", "MID" };
char const** cp[] = { c + 2, c + 3, c , c + 1 };
char const*** cpp = cp;
int main()
{
	cout << *cpp[1] << endl;
	cout << *(*(*(cpp + 2) + 2) + 3) << endl;
	cout << (*cpp)[-1] << endl;
	cout << *(cpp + 3)[-1] << endl;
```

```
        return 0;
}
```

Question – 4:

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

main()
{
   int ary[2][2][4] = { {{1,2,3,5},{3,4,6,7}},{{5,6,5,1},{7,8,2,4}} };
   int(*p)[2][4] = (int(*)[2][4])ary;

   cout << *(*(*(p + 2) + 1) + 1);
   for (int i = 0; i < 2; i++)
   {
      for (int j = 0; j < 2; j++)
      {
         for (int k = 0; k < 2; k++)
         {
            cout << *(*(*(p + i) + j) + k) << "\t";
         }
         cout << "\n";
      }
      cout << "\n\n\n";
   }
}
```

Question – 5:

```
int main()
{
        int array[2][5][2] = { 10,20,30,40,50,60,70,
                                        80,90,100,18,21,3,4,
                                        5,6,7,81,9,11 };
        int(*p)[5][2];
        p = array;
        for (int i = 0; i < 2; i++)
                cout << "\nthe vale is " << *((int*)(p + 1) + (1 * 2) + i);
        return 0;
}
```

Question – 6:

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

main()
{
   int ary[2][6] = { {2,5,6,4,9,1},{7,8,12,11,32,11} };
   int(*ptr)[2] = (int(*)[2])ary + 3;

   for (int i = 0; i < 2; i++)
   {
      for (int j = 0; j < 3; ++j)
      {
```

```
            cout << *(*(ptr + i) + j) << "\t";
        }
        cout << endl;
    }
}
```

Question – 7:

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

int main()
{
    const char* str[] = { "AAAAA", "BBBBB", "CCCCC", "DDDDD" };
    const char** sptr[] = { str + 3, str + 2, str + 1, str };
    const char*** pp;

    pp = sptr;
    ++pp;
    cout << **++pp + 2;

}
```

Question – 8:

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

main()
{
        int* ip = new int;
        short* sp;
        char* cp;

        *ip = 16706;                    //Hex 4142
        *ip=65;
        //cp=ip;

        cp = (char*)ip;
        cout << *cp << endl;
        cout << *(cp + 1) << endl;

        sp = (short*)ip;
        cout << *sp;

}
```

Question – 9:

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

void foo(int(*ptr)[4]) {
    cout << ptr[0][0] << "  ";
```

```
}
int main()
{
   int arr[9] = { 2,4,6,8,10,12,114,16,18 };

   foo((int(*)[4])(arr));

   foo((int(*)[4])(arr + 2));

   foo((int(*)[4])(arr) + 1);

   int arr2[3][4] = { 12,24,36,48,60,72,84,96,108,120,132,144 };

   foo((int(*)[4])(arr2) + 1);

   return 0;
}
```

Question – 10:

```
int print_row(int ct, int num)
{
        if (num == 0)
                return ct;
        cout << ct << "\t";
        print_row(ct + 1, num - 1);
}
void pattern(int n, int count, int num)
{
        if (n == 0)
                return;
        count = print_row(count, num);
        cout << endl;
        pattern(n - 1, count, num + 1);
}
int main()
{
        int n = 5;
        pattern(n, 1, 1);
        return 0;
}
```

Question – 11:

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

void find(int, int&, int&, int = 4);
int main() {
```

```
	int one = 1, two = 2, three = 3;
	find(one, two, three);
	cout << one << " " << two << " " << three << endl;
	return 0;
}
void find(int a, int & b, int& c, int d) {
	if (d < 1)
		return;
	cout << a << " " << b << " " << c <<endl;
	c = a + 2 * b;
	int temp = b;
	b = a;
	a = 2 * temp;
	d % 2 ? find(b, a, c, d - 1) : find(c, b, a, d - 1);
}
```

Question – 12:

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

int fun(int n, int* fp)
{
   int t, f;

   if (n <= 2) {
      *fp = 1;
      return 1;
   }
   t = fun(n - 1, fp);
   f = t + *fp;
   *fp = t;
   return f;
}

int main()
{
   int x = 15;
   cout << fun(5, &x) << endl;
   return 0;
}
```

Question – 13:

```
#include <iostream>
using namespace std;
class Dummy {
	float z;
	int x, y;
public:
	Dummy(int x = 0, int y = 1) :x(x + 2), y(y + 3) {
```

```
                z = x + y + 1;
        }
        void print() {
                cout << " X= " << x
                        << " Y = " << y
                        << " Z = " << z;
        }
};

int main() {

        Dummy d(10);
        d.print();
}
```

Question – 14:

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


class A {

public:
   A(int ii = 0) : i(ii), s(new int{ i + 1 }) {}
   A(const A& abc)
   {
      this->i = abc.i;
      this->s = new int(*(abc.s));
      cout << "Out Of " << i + *s << endl;
   }
   A magic(A abc) {
      A bcd(2);
      return abc;
   }

   ~A() { cout << "Out A " << i << endl; }

private:
   int i;
   int* s;
};

int main() {

   A b(3), a(4);
   a = b.magic(a).magic(b);

}
```

Question – 15:

```
#include<iostream>
using namespace std;
class mystery
{
private:
   int* n;
public:
   mystery() :n(new int)
   {
      *n = 5;
   }
   mystery(int nn) :n(new int) {
      *n = nn;
   }
   mystery& operator=(const mystery& n)
   {
      this->n = new int;
      *this->n = *n.n;
      return *this;
   }
   mystery& display()
   {
      cout << *n << " ";
      return *this;
   }
   void increase()
   {
      *n += 1;
   }
};

int main() {

   mystery b(1), c = b, d;
   b.increase();
   d = b = c;
   mystery a(d);
   a.increase();
   c.increase();
   a.display().display();
   mystery l = b.display();
   c.display();
   d.display();
   l.display();

}
```

Question - 16:

```
#include <iostream>
using namespace std;
class A
{

   int data[2];
   int arr[3];
   int ss;
public:
   A(int x, int y) : data{ x, y }, arr{ x + y, y - x, y % x }
```

```
    {
        ss = y / x;
    }

    A(int* ptr) : data{ *ptr, *(ptr + 1) }, arr{ 0 }
    {
        ss = *ptr;
    }

    void display() { cout << data[1] + ss + arr[2] << endl; }

    ~A() { cout << arr[0] - data[0] - ss << endl; }
};

int main()
{
    A a(22, 33);

    int* arr = (int*)&(a);
    arr += 3;
    cout << arr[-2] + arr[2] << endl;
    a = (arr - 2);
    a.display();

}
```

Question – 17:

```
#include<iostream>
using namespace std;
class mystery
{
private:
    int* n;
public:
    mystery() :n(new int)
    {
        *n = 5;
    }
    mystery(int nn) :n(new int) {
        *n = nn;
    }
    mystery& operator=(const mystery& n)
    {
        this->n = new int;
        *this->n = *n.n;
        return *this;
    }
    mystery display()
    {
        cout << *n << " ";
        return *this;
    }
    int increase()
    {
        *n += 1;
        return *n;
    }
    ~mystery()
```

```
    {
        cout << "Bye " << *n << endl;
    }
};

int main() {

    mystery b(1), c = b, d;
    b = c = d;
    mystery a(d);
    a.increase();
    a.display().increase();
    mystery l = b.display().increase();
    c = 10;
    c.display().increase();
}
```

Question - 18:

```
#include<iostream>
using namespace std;
class mystery
{
private:
    int* n;
public:
    mystery() :n(new int)
    {
        *n = 5;
    }
    mystery(int nn) :n(new int) {
        *n = nn;
    }
    mystery(const mystery& n)
    {

        this->n = new int;
        *this->n = *n.n;
    }
    mystery display()
    {
        cout << *n << " ";
        return *this;
    }
    int increase()
    {
        *n += 1;
        return *n;
    }

    operator int()
    {
        return *n + 3;
    }

    int& operator() (int ss)
    {
        *n += ss;
        return *n;
```

```
    }
    ~mystery()
    {
        cout << "Bye " << *n << endl;
    }
};

int main() {

    mystery b(1), c = b, d;
    b = c = d;
    mystery a(d);
    a.increase();
    a.display().increase();
    mystery l = b.display().increase();
    l.display().increase();

    int* ptr = new int(c);
    cout << *ptr;
    delete ptr;
    c(5) = 3;
    c.display();

}
```

Question - 19:

```
#include<iostream>
using namespace std;
class mystery
{
private:
    int* n;
    int arr[3];

public:
    mystery() :n(new int(5)), arr{ *n, *n + 1, *n + 2 }
    {
        (*this)(4, 2) = 8;
    }
    mystery(int nn) :n(new int), arr{ nn, nn + 1, nn + 2 }
    {
        *n = nn;
    }
    mystery(const mystery& n)
    {

        this->n = new int;
        *this->n = *n.n;
        this->arr[0] = n.arr[0];
        this->arr[1] = n.arr[1];
        this->arr[2] = n.arr[2];

    }
    mystery display()
```

```
    {
        cout << *n << " " << arr[0] << " ";
        return *this;
    }
    int increase()
    {
        *n += 1;
        return *n;
    }

    operator int()
    {
        return *n + 3;
    }

    int& operator() (int ss, int pr)
    {
        *n += ss;
        return this->arr[ss - pr];

    }
    ~mystery()
    {
        cout << "Bye " << *n + arr[1] << endl;
    }
};

int main() {

    static mystery b(1), c = b, d;
    b = c = d;
    mystery a(d);
    a.increase();
    a.display().increase();
    mystery l = b.display().increase();
    l.display().increase();
    static mystery s = l.increase();

    s(5, 3) = 6;
    b(4, 3) = 1;
    a(8, 8) = 7;

}
```

Question - 20:

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

class magic
{
    int s;
public:
    magic(int ss) : s(ss + 2) { };

    int do_magic()
```

```
    {
        cout << "MAGIC " << s << endl;
        return s;
    }
    ~magic()
    {
        cout << "No MAGIC " << s << endl;
    }

};
class mystery
{
private:
    int* n;
    int arr[3];

public:
    mystery() :n(new int(5)), arr{ *n, *n + 1, *n + 2 }
    {   }

    mystery(int nn) :n(new int), arr{ nn, nn + 1, nn + 2 }
    {
        *n = nn;
    }

    mystery(const mystery& n)
    {

        this->n = new int;
        *this->n = *n.n;
        this->arr[0] = n.arr[0];
        this->arr[1] = n.arr[1];
        this->arr[2] = n.arr[2];

    }

    int& operator() (int ss, int pr)
    {
        *n += ss;
        return this->arr[ss - pr];
    }
    magic* operator->()
    {
        static int s = 2;
        magic* m = new magic(s);
        s++;
        return m;
    }

    mystery operator++()
    {
        *n += 1;
        return *n;
    }

    void smile(int a)
    {
        cout << (*this)(7, a) << endl;
    }
```

```
    ~mystery()
    {
        cout << "Bye " << *n + arr[1] << endl;
    }

    friend ostream& operator<<(ostream& out, const mystery& m);
};

ostream& operator<<(ostream& out, const mystery& m)
{
    out << *m.n << " " << m.arr[0] << endl;
    return out;
}

int main() {

    static mystery b(1), c = b, d;
    mystery* monster = new mystery(5);
    mystery a(d);
    ++a;

    mystery l = d->do_magic();

    monster->smile((*monster)->do_magic());

    b(4, 3) = l->do_magic();
    a(8, 8) = a->do_magic();

    cout << a << b << l;

}
```

Question - 21:

```
#include<iostream>
using namespace std;
class Num {
    int* n;
    static int c;
public:
    Num() :n(new int) {
        *n = 4;
    }
    Num(int* nn) :n(nn) {
        c++;
        cout << *n << " " << c << endl;
    }
    Num(Num& otherNum) :n(otherNum.n) {
        cout << *n << " " << endl;
        *n += 4;
        c++;
    }
```

```
    void display()const {
        cout << *n << "" << endl;
    }
    void display(Num n)const {
        *n.n = +1;
        cout << *(this->n) << "" << endl;
    }
    ~Num() {
        cout << c << " " << *n << endl;
        --c;
    }
};
int Num::c = 0;

int main() {
    Num a; int n = 8;
    Num b(&n);
    const Num c(a);
    c.display();
    a.display(b);
    cout << "--------" << endl;
}
```

Question – 22:

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

class A {
    int x;
public:
    A(int a) :x(a) {
        cout << x << endl;
    };
    ~A() {
        cout << x << endl;
    }

};
A a(2);
int main(int argc, char* argv[])
{
    static A b(3);
    {
        A c(4);
    }

}
```

Question – 23:

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

class Mystery
```

```
{
public:
   static int n;

   Mystery()
   {
      cout << n++ << endl;
   }
   Mystery(int i)
   {
      n = i;
      cout << n << endl;
   }
   static void somefunc()
   {
      n = 5;
   }

   Mystery(Mystery const& otherNum)
   {
      n += 5;
      cout << n << endl;
   }

   ~Mystery()
   {
      cout << --n << "\n";
   }
}a;

void fun(Mystery n)
{
   cout << n.n << endl;
   n.somefunc();
}

int Mystery::n = 0;

int main()
{
   Mystery b(9), c;
   fun(b);
}
```

Question – 24:

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

class Complex
{
   double r, i;

public:
   Complex(double r = 1.0, double i = 1.0)
```

```
    {
        set(r, i);
    }

    void set(double r, double i)
    {
        Complex::r = r;
        this->i = i;
    }


    void print()
    {
        if (i < 0)
            cout << r << "" << i << "i" << endl;
        else
            cout << r << "+" << i << "i" << endl;
    }

    Complex operator+(Complex R)
    {
        Complex tmp;
        tmp.r = r + R.r;
        tmp.i = i + R.i;
        return tmp;
    }

    Complex operator++()
    {
        Complex tmp = *this;
        r++;
        i++;
        return tmp;
    }

    Complex operator++(int)
    {
        ++(*this);
        return *this;
    }
};

int main()
{
    Complex A(3, 4), B(5, -6);
    A.print();
    B.print();
    Complex C;
    C = A + B;
    C.print();
    (++A).print();
    C = ++A;
    C.print();
    (A++).print();
    A.print();

}
```

Question – 25:

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

class Point
{
   int x, y;

public:
   Point(int x = 0, int y = 0)
   {
      this->x = x;
      Point::y = y;
      (*this)();
   }

   void operator()()
   {
      cout << " (" << x << ", " << y << ") " << endl;
   }

   Point& operator()(int y)
   {
      this->y = y;
      return *this;
   }

   ~Point()
   {
      cout << "Point is going";
      (*this)();
   }
}p3;

int main()
{
   Point* p = new Point(5, 6);
   static Point p1(p3);
   p1(9)(8);
   delete p;
   Point p2(7);
   cout << "----------" << endl;
}
```

Question – 26:

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

class ItsMagic {
public:
   int* value;
```

```
    ItsMagic(int n = 8) : value(new int[n - 5] {n})
    {
        for (int i = 0; i < n - 7; i++)
            *(value + i + 1) = *(value + i) + i + 3;
        cout << "Hello <:> " << value[2] << endl;
    }

    ItsMagic(const ItsMagic& oh)
    {
        this->value = oh.value + 1;
        *this->value = *oh.value + 5;
        (*this)(oh.value + 1);
        cout << "Oh Ho <:> " << value[2] << endl;
    }

    int& operator()(int* a)
    {
        *(this->value + 2) = *a++;
        cout << "Is it you -): " << *this->value << endl;
        return *(this->value + 1);
    }

    void increase(int& n)
    {
        static int N = 5;
        n = N++;
        if (n % 3 == 2)
            this->twice(N);
        cout << "Seriously -> " << N << endl;
    }

    void twice(int& n)
    {
        static int N = 6;
        n = ++N;
        if (n % 4 == 0)
            this->increase(N);
        cout << "Please -> " << N << endl;
    }

    ~ItsMagic()
    {
        int s = 3;
        cout << "Don't..... ";
        this->increase(s);
        cout << s << endl;
    }

};

class NoWay {
public:
    ItsMagic okay;
    int s;
    NoWay(int a) :okay(a)
    {
        s = *okay.value + 3;
        cout << *(okay.value + 2) << endl;
```

```
        cout << "Its Okay :) " << okay(okay.value) << endl;
    }

    ItsMagic& neverMind()
    {
        okay.increase(s);
        cout << "Never Mind :( " << s + okay(okay.value + 1) << endl;
        return okay;
    }

    ~NoWay()
    {
        int sum = 0;
        cout << "Are you going ? \n";
        for (int i = 0; i < 3; i++)
            sum += okay.value[i];
        cout << "Here take this -> " << sum << endl;
    }

};

void comeHere(ItsMagic boo)
{
    boo(boo.value);
    cout << "Bye :( " << *boo.value++ << endl;

}

int main()
{
    ItsMagic isIt;
    NoWay areYou(10);
    comeHere(areYou.neverMind());

}
```

Question – 27:

```
#include <iostream>
using namespace std;
class Point {
    int x, y;
public:
    Point(int a = 0, int b = 0)
    {
        x = a;
        y = b;
        print();
    }
    void print()
    {
        cout << " (" << x << "," << y << ") " << endl;
    }
    ~Point()
```

```
    {
        cout << "Point is going" << endl;
    }

};
class Circle
{
    Point center;
    float radius;
public:
    Circle() :center(0, 0)
    {
        radius = 0;
        cout << "The basic circle" << endl;
    }
    Circle(Point p) :center(p) { }

    Circle(const Circle& c) :center(c.center), radius(c.radius)
    {
        cout << "The copied circle";
        center.print();
    }
    ~Circle()
    {
        cout << "Circle is going" << endl;
    }

};

int main()
{
    Point p1;
    Circle c1;
    static Circle c2(p1);
    Circle c3(c2);

}
```

Question – 28:

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

class Engine {
public:
    int cylinders;

    Engine(int numCylinders) : cylinders(numCylinders) {
        cout << "Creating Engine with " << cylinders << " cylinders" << endl;
    }
    ~Engine() { cout << "Destroying Engine with " << cylinders << " cylinders" << endl; }

};
```

```
class Car {
public:
    Engine engine;
    string make;
    string model;

    Car(const string& carMake, const string& carModel, int numCylinders)
        : engine(numCylinders), make(carMake), model(carModel) {
        cout << "Creating " << make << " " << model << " with " << numCylinders << " cylinders" << endl;
    }
    ~Car() { cout << "Destroying " << make << " " << model << " with " << engine.cylinders << " cylinders" << endl; }
};

class Person {
public:
    string name;

    Person(const string& personName) : name(personName) {
        cout << "Creating Person named " << name << endl;
    }
    ~Person() { cout << "Destroying Person named " << name << endl; }
};

class Driver {
private:
    Person person;
    Car car;
public:
    Driver(const string& driverName, const string& carMake, const string& carModel, int numCylinders)
        : person(driverName), car(carMake, carModel, numCylinders) {
        cout << "Creating Driver named " << driverName << " with " << carMake << " " << carModel << " with " <<
numCylinders << " cylinders" << endl;
    }
    ~Driver() { cout << "Destroying Driver named " << person.name << " with " << car.make << " " << car.model << "
with " << car.engine.cylinders << " cylinders" << endl; }
};



int main() {

    Car myCar("Honda", "Civic", 4);

    Person myPerson("Alice");
    Driver myDriver("Bob", "Toyota", "Corolla", 4);

    {
        Driver myDriver("Charlie", "Ford", "Mustang", 8);
    }

}
```

Question - 29:

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

```
class A
{
private:
	int a;
public:
	A(int x = 10) { a = x; cout << "A() called for " << a << " .\n"; }
	~A() { cout << "~A() called for a = " << a << endl; }
	void Print() { cout << "a = " << a << endl; }
};
class B
{
private:
	int b;
	A a;
	A* aptr;
public:
	B() { b = 0; aptr = 0; cout << "B() called." << endl; }
	B(int x, A* objPtr) :a(x + 5)
	{
		b = x;
		aptr = objPtr;
		cout << "B() called for b = " << b << endl;
	}
	void Print() {
		cout << "b = " << b << endl; a.Print();
		if (aptr != 0) aptr->Print();
	}
	~B() { cout << "~B() called for b = " << b << endl; }
};
int main()
{
	A a1(5);
	B b1(10, &a1);
	cout << "--------------------------\n";
	b1.Print();
}
```

Question – 30:

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

class Number {
public:
	int* value;
	Number(int v) {
		value = new int(v);
		cout << "Value: " << *value << endl;
	}
	~Number() {
		cout << "Killed: " << *value << endl;
		delete value;
	}
};
class Question {
public:
```

```
        Number marks;
        Question(int A) : marks(A) {
                cout << "New Object \n";
        }
        Question(const Question& X) : marks(*X.marks.value + 10) {
                cout << "ItsEasy" << endl;
        }
};
void Difficult(Question why) {
        Question Quest = why;
}
int main() {
        Question Answer(1);
        Difficult(Answer);
}
```

Question – 31:

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

class XYZ
{
private:
   int x;

public:
   XYZ(int y = 10)
   {
      x = y;
      cout << "XYZ() called for " << x << endl;
   }
   void Print()
   {
      cout << x << endl;
   }
   ~XYZ()
   {
      cout << "~XYZ() Called.\n";
   }
};

class ABC
{
   int c;

public:
   XYZ a;
   XYZ* b;
   ABC(int val = 50)
   {
      c = val;
      cout << "ABC() called for " << c << endl;
      b = new XYZ(a);
   }
   void Print()
   {
      cout << "c = " << c << endl;
      cout << "a = ";
```

```
        a.Print();
        if (b != nullptr)
        {
            cout << "b =  ";

            b->Print();
        }
    }
};

int main()
{
    ABC* x = new ABC;
    x->Print();
    XYZ* ptr = &(x->a);
    delete x;
    ptr->Print();
}
```

Question – 32:

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

class Complex {
private:
        double real;
        double imag;

public:

        Complex(double r = 0.0, double i = 0.0) : real(r), imag(i)
        { }

        bool operator == (Complex rhs)
        {
                return (real == rhs.real &&
                        imag == rhs.imag);
        }
};

int main()
{
        Complex com1(3.0, 0.0);

        if (com1 == 3.0)
                cout << "Same";
        else
                cout << "Not Same";
        return 0;
}
```

Question – 33:

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

```
class fun {
private:
	int x;
public:
	fun(int x1 = 0) {
		x = x1;
		cout << "constructor of ";
		print();
	}
	int getX() { return x; }
	void setX(int x1) { x = x1; }
	void print() {
		cout << "(" << x << ")" << endl;
	}
	fun(const fun& obj) {
		x = obj.x;
		cout << "Copy constructor of ";
		print();
	}
	~fun() {
		cout << "destructor of ";
		print();
	}

};
void print(const int* p, int n) {
	for (int i = 0; i < n; i++)
		cout << p[i] << " ";
	cout << endl;
}
int main()
{
	cout << "Output (if any): " << endl;
	fun a(6);
	int list[6] = { 0,10,20,30,40,50 };
	int length = 3;
	int* array = &length;
	int* p = list;
	fun b = function(array, p, a);
	cout << "content of array: ";
	print(array, a.getX());
	cout << "content of p: ";
	print(p, (length / 2));
	cout << "content of list: ";
	print(list, length);
	cout << "Output (if any): " << endl;

}
```

Question – 34:

```
#include <iostream>

class Base {
public:
   virtual void sayHello() {
      std::cout << "Hello world, I am Base" << std::endl;
   }
```

```
};

class Derived : public Base {
public:
   void sayHello() {
      std::cout << "Hello world, I am Derived" << std::endl;
   }
};

void testPointer(Base* obj) {
   obj->sayHello();
}

void testReference(Base& obj) {
   obj.sayHello();
}

void testObject(Base obj) {
   obj.sayHello();
}

int main() {
   {
      std::cout << "Testing with pointer argument: ";
      Derived* derived = new Derived;
      testPointer(derived);
   }
   {
      std::cout << "Testing with reference argument: ";
      Derived derived;
      testReference(derived);
   }
   {
      std::cout << "Testing with object argument: ";
      Derived derived;
      testObject(derived);
   }
}
```

Question – 35:

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

class Vehicle
{
public:
   Vehicle()
   {
      cout << "Vehicle() called.\n";
   }
   ~Vehicle()
   {
      cout << "~Vehicle() called.\n";
   }
   virtual void Print()
   {
      cout << "Test\n";
   }
```

```
};
class MotorCycle : public Vehicle
{
public:
   MotorCycle()
   {
      cout << "MotorCycle() called.\n";
   }
   ~MotorCycle()
   {
      cout << "~MotorCycle() called.\n";
   }
};
class Car : public Vehicle
{
public:
   Car()
   {
      cout << "Car() called.\n";
   }
   ~Car()
   {
      cout << "~Car() called.\n";
   }
   virtual void Print()
   {
      cout << "Check\n";
   }
};
int main()
{
   Vehicle* vehicles[3];
   vehicles[0] = new MotorCycle;
   vehicles[1] = new Car;
   vehicles[2] = new Vehicle;
   for (int i = 0; i < 3; i++)
      vehicles[i]->Print();
   for (int i = 0; i < 3; i++)
      delete vehicles[i];
}
```

Question – 36:

```
//----------------------Syntax Errors----------------------//
//-----------Do not Run only point out Errors-------------//

 class D
 {
    int y;
    void walk()
    {
       cout << "walk of D" << endl;
    }

 public:
    D(int y1 = 0)
    {
       y1 = y;
    }
```

```
}

class A
{

public:
   int x;
   void print()
   {
      cout << "----A----        x:" << x << endl;
   }
   A(int x1 = 0)
   {
      x = x1;
   }
};
class B : A
{
   D x;

public:
   D getx()
   {
      return x;
   }
   virtual void print() = 0;
   B(int x1, int y1) : D(y1), A(x1)
   {
   }
};
class C : B
{
public:
   int x;
   C(int x1 = 0, int x2 = 10, int x3 = 20) : B(x1, x2)
   {
      x = x3;
   }
   void print()
   {
      cout << "----C----        x:" << x << endl;
      A::print();
      B::print();
   }
   void fun()
   {
      cout << "its fun" << endl;
   }
};

int main()
{
   B *p = new B;

   A *q = new A;

   q->print();

   q->A();
```

```
    B *ptr = new C;

    ptr->x = 35;

    ptr->print();

    ptr->getx().walk();

    C *p1 = dynamic_cast<C *>(ptr);

    (p1->fun()).fun();


}
```

Question – 37:

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


class D
{
public:
   D() { cout << "D ctor" << endl; }
   D(D&) { cout << "D copy ctor" << endl; }
   ~D() { cout << "D dtor" << endl; }
};
class A
{
public:
   A() { cout << "A ctor" << endl; }
   ~A() { cout << "A dtor" << endl; }
};
class B : public A
{
public:
   B() { cout << "B ctor" << endl; }
   ~B() { cout << "B dtor" << endl; }
   void test(D d) { A a; }
};
B globalB;

int main()
{
   A a;
   D d;
   D d2 = d;
   d = d2;
   globalB.test(d);
}
```

Question – 38:

```
class A
```

```
{
public:
   A() { cout << "In As constructor" << endl; }
   ~A() { cout << "In As destructor" << endl; }
};


class B : public A
{
public:
   B() { cout << "In Bs constructor" << endl; }
   ~B() { cout << "In Bs destructor" << endl; }
};

class C : public B
{
public:
   C() { cout << "In Cs constructor" << endl; }
   ~C() { cout << "In Cs destructor" << endl; }
};


int main()
{
   C x1;
   C * x2  = new C;
}
```

Question – 39:

```
//----------------------Question # 5----------------------//
class A
{
   int a;

public:
   A() : a(0)
   {
      cout << "A()" << endl;
   }
   A(int a) : a(a)
   {
      cout << "A(int a)" << endl;
   }
   ~A()
   {
      cout << "~A()" << endl;
   }
   void print()
   {
      cout << "a=" << a << endl;
   }
   void seta(int a)
   {
      this->a = a;
      cout << "seta(int a)" << endl;
   }
```

```
protected:
   void prot_func_A()
   {
      cout << "prot_func_A()from A" << endl;
   }
};
class B : protected A
{
   int b;

public:
   B() : b(0)
   {
      cout << "B()";
   }
   B(int b, int a = 0) : b(b)
   {
      cout << "B(intb, inta=0)" << endl;
      seta(a);
   }
   ~B()
   {
      cout << "~B()" << endl;
   }
   void print()
   {
      cout << "b=" << b << endl;
      A::print();
   }
   void prot_func_A()
   {
      cout << "prot_func_A() fromB" << endl;
   }
};

int main()
{
   B objB(10, 20);
   objB.print();
   objB.prot_func_A();
}
```

Question - 40:

```
class Yes
{
public:
   Yes() { cout << " Yes() "; }
};

class No
{
   Yes y;
public:
   No() { cout << " No() "; }
};

class Emotion {
```

```
public:
   Emotion()
   {
      cout << "Emotion() ";
   }
};

class Sad : public Emotion
{
   No n;

public:
   Sad() { cout << "Sad() "; }
};

class Depress : public Sad
{
public:
   Depress() {
      cout << " Depress() ";
   }
};

int main()
{
   Depress why;
   cout << "\nOH No! :( \n";
   cout << "OH Yeah! ): \n";
   Sad noWay;
}
```

Question - 41:

```
class Parent
{
   int* b;

public:
   Parent() { b = new int(10); }
   virtual void Print()
   {
      cout << "B = " << *b << endl;
   }
   ~Parent() { delete b; }
};

class Child : public Parent
{
   int* d;

public:
   Child() { d = new int(20); }
   void Print()
   {
      Parent::Print();
      cout << "D = " << *d << endl;
   }
   ~Child() { delete d; }
```

```
};

int main()
{
   Parent* pPtr = new Child();
   pPtr->Print();
   delete pPtr;
}
```

Question - 42:

```
class B
{
private:
   int* bptr;

public:
   B(int b = 10) { bptr = new int(b); }
   virtual int GetValue()
   {
      return *bptr;
   }
   virtual ~B()
   {
      cout << "~B()";
      if (bptr != 0)
         delete bptr;
   }
};

class D1 : public B
{
private:
   int* dptr1;

public:
   D1(int d1 = 20) { dptr1 = new int(d1); }
   int GetValue()
   {
      return (B::GetValue() + *dptr1);
   }
   void Print()
   {
      cout << "*dptr1 = " << *dptr1 << endl;
   }
   ~D1()
   {
      cout << "~D1()";
      if (dptr1 != 0)
         delete dptr1;
   }
};

class D2 : public B
{
private:
   int* dptr2;
```

```
public:
   D2(int d2 = 30)
   {
      dptr2 = new int(d2);
   }
   int GetValue()
   {
      return (B::GetValue() + *dptr2);
   }
   ~D2()
   {
      cout << "~D2()";
      if (dptr2 != 0)
         delete dptr2;
   }
};

class GC : public D1
{
private:
   int* gcPtr;

public:
   GC(int gc = 40) : D1(gc + 10)
   {
      gcPtr = new int(gc);
   }
   int GetValue()
   {
      return (D1::GetValue() + *gcPtr);
   }
   void Print()
   {
      cout << "*gcptr = " << *gcPtr << endl;
   }
   ~GC()
   {
      cout << "~GC()";

      if (gcPtr != 0)
         delete gcPtr;
   }
};
int main()
{
   B* arr[4];
   arr[0] = new B(1);
   arr[1] = new D1(2);
   arr[2] = new D2(3);
   arr[3] = new GC(4);

   for (int i = 0; i < 4; i++)
   {
      cout << arr[i]->GetValue() << " , ";
   }

   cout << endl;

   for (int i = 0; i < 4; i++)
```

```
    {
        delete arr[i];
        cout << endl;
    }

    cout << "-------------------------\n";

    D1* arr2[2];
    arr2[0] = new D1(100);
    arr2[1] = new GC(500);

    for (int i = 0; i < 2; i++)
        arr2[i]->Print();

    for (int i = 0; i < 2; i++)
    {
        delete arr2[i];
        cout << endl;
    }
}
```

Question – 43:

```
class ThermalReactor
{
    int valve;
    float temprature;

public:
    ThermalReactor(int v, float t)
    {
        valve = v;
        temprature = t;
    }
    virtual void print()
    {
        cout << "Valve: " << valve;
        cout << " Temparture:" << temprature << endl;
    }
};

class MagnoxReactor : public ThermalReactor
{
    float maxPower;
    float production;

public:
    MagnoxReactor(int v, float t, float m, float p) : ThermalReactor(v, t)
    {
        maxPower = m;
        production = p;
    }

    bool isAtCritical()
    {
        return (maxPower == production);
    }
```

```
    void signal()
    {
        cout << "Production cannot be increased" << endl;
    }

    void increaseProd(float factor)
    {
        if ((production + factor) < maxPower)
        {
            production += factor;
            print();
        }
        else
            signal();
    }

    void print()
    {
        ThermalReactor::print();
        cout << "Current production: " << production;
        cout << " Max Power: " << maxPower << endl;
    }
};

void Capacity(ThermalReactor* reactor)
{
    reactor->print();
    dynamic_cast<MagnoxReactor*>(reactor)->increaseProd(10);
}

int main()
{
    MagnoxReactor* MagRec = new MagnoxReactor(4, 1000, 330, 200);
    Capacity(MagRec);
    return 0;
}
```

Question – 44:

```
class A
{
public:
    int f() { return 1; }
    virtual int g() { return 2; }
};

class B : public A
{
public:
    int f() { return 3; }
    virtual int g() { return 4; }
};

class C : public A
{
public:
    virtual int g() { return 5; }
};
```

```
int main()
{
   A* pa;
   A a;
   B b;
   C c;
   pa = &a;
   cout << pa->f() << endl;
   cout << pa->g() << endl;
   pa = &b;
   cout << pa->f() + pa->g() << endl;
   pa = &c;
   cout << pa->f() << endl;
   cout << pa->g() << endl;
   return 0;
}
```

Question - 45:

```
class A
{
private:
   int* aptr;

public:
   A(int a = 5)
   {
      aptr = new int(a);
   }
   virtual void Print()
   {
      cout << "a = " << *aptr << endl;
   }
   virtual ~A()
   {
      cout << "~A() called.\n";
      if (aptr != 0)
         delete aptr;
   }
};

class B : public A
{
private:
   int* bptr;

public:
   B(int b = 10)
   {
      bptr = new int(b);
   }
   void Print()
   {
      A::Print();
      cout << " b = " << *bptr << endl;
   }
```

```
    ~B()
    {
        cout << "~B() called.\n";
        if (bptr != 0)
            delete bptr;
    }
};

class C : public A
{
private:
    int* cptr;

public:
    C(int c = 15) : A(c * 10)
    {
        cptr = new int(c);
    }
    void Print()
    {
        A::Print();
        cout << " c = " << *cptr << endl;
    }
    ~C()
    {
        cout << "~C() called.\n";
        if (cptr != 0)
            delete cptr;
    }
};

int main()
{
    A* aptr[3];
    aptr[0] = new B(1);
    aptr[1] = new C(2);
    aptr[2] = new A(3);

    for (int i = 0; i < 3; i++)
        aptr[i]->Print();

    for (int i = 0; i < 3; i++)
        delete aptr[i];
}
```

Question - 46:

```
class StudentInfo
{
public:
    StudentInfo() {
        cout << "StudentInfo() called" << endl;
    }

    ~StudentInfo() {
        cout << "Dest-StudentInfo() called" << endl;
    }
```

```
};

class Students
{
public:
   Students() {
      cout << "Students() called" << endl;
   }

   StudentInfo info;

   ~Students() {
      cout << "Dest-Students() called" << endl;
   }

};

class AcademicInstitutions
{
public:
   AcademicInstitutions() {
      cout << "AcademicInstitutions() called" << endl;
   }

   ~AcademicInstitutions() {
      cout << "Dest-AcademicInstitutions() called" << endl;
   }

};

class Universities : public AcademicInstitutions
{
   Students Aneeq;

public:
   Universities() {
      cout << "Universities() called" << endl;
   }

   ~Universities() {
      cout << "Dest-Universities() called" << endl;
   }

};

class PrivateUniversity : public Universities
{
public:
   PrivateUniversity() {
      cout << "PrivateUniversity() called" << endl;
   }
   ~PrivateUniversity() {
      cout << "Dest-PrivateUniversity() called" << endl;
   }
};

int main()
```

```
{

   AcademicInstitutions* inst = new PrivateUniversity;
   delete inst;
}
```

Question - 47:

```
//-----Cross out all the lines that will not compile------//
//------in the main function of the following program------//
//----------------Do NOT run on Compiler--------------------//

class A
{
public:
   A() {}
   virtual int output() = 0;

private:
   int i;
};

class B : public A
{
private:
   int j;
};

class C
{
public:
   int f(int a) { return x * a; }

protected:
   void setX(int a) { x = a; }
   int getX() { return x; }

private:
   int x;
};

class D : public C
{
private:
   int z;
};

int main()
{
   A objA;
   B objB;
   C objC;
   D objD;
   C.setX(2);
   cout << C.getX();
   D.setX(1);
   D.f(3);
   return 0;
}
```

Question – 48:

```
class M
{
public:
   virtual void myMemory()
   {
      cout << "I forget ";
   }
   void Disk()
   {
      cout << "Space ";
   }
   void Erased()
   {
      cout << "For good ";
   }
   void thisExam()
   {
      Erased(); myMemory();
   }
   virtual ~M() {}

};

class N : public M
{
public:
   void myMemory()
   {
      cout << "Gone ";
   }
   void Disk()
   {
      cout << "Slipped ";
   }

   void virtual Erased()
   {
      cout << "Rubbed out ";
   }
};

int main()
{
   M* m1 = new N;
   m1->myMemory();
   m1->Disk();
   m1->thisExam();

   M m2 = *(new N);
   m2.myMemory();
   m2.Disk();
   m2.thisExam();

}
```

Question – 49:

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

class S
{
	int s;
public:
	S(int s = 0) : s(s) { cout << "H S " << this->s << endl; }
	void print()
	{
		cout << s << endl;
	}
	~S()
	{
		cout << "Bye S" << endl;
	}
};

class A : virtual public S
{
	int a;
public:
	A(int s = 0) : a(s) { cout << "H A " << a << endl; }
	void print()
	{
		cout << a << endl;
	}
	~A()
	{
		cout << "Bye A" << endl;
	}
};

class B : virtual public A
{
	int b;
public:
	B(int n = 0) :A(9)
	{
		b = n;
		cout << "HB " << b << endl;
	}
	void display()
	{
		cout << b << endl;
	}
	~B()
	{
		cout << "bye B" << endl;
	}
};

class C :virtual public A
{
	int c;
public:
```

```
	C(int n = 0)
	{
		c = n;
		cout << "HC " << c << endl;
	}
	void display()
	{
		cout << c << endl;
	}
	~C()
	{
		cout << "bye C" << endl;
	}
};

class D : public C, public B
{
	int d;
public:
	D(int n = 0) :C(3), B(2), A(4), S(2)
	{
		d = n;
		cout << "H D " << d << endl;
	}
	void display()
	{
		cout << d << endl;
	}
	~D()
	{
		cout << "bye D" << endl;
	}
};

int main()
{
	D obj;
	//	obj.display();


}
```

Question – 50:

```
#include <iostream>
using namespace std;
class Course
{
public:
   virtual void Pro1() { cout << "Prioritize "; }
   virtual void Pro2() { cout << "your "; }
   virtual void Pro3() { cout << "work "; }
   virtual void Con1() { cout << "not fun "; }
};

class Programming : public Course
```

```
{
public:
   virtual void Pro1() { cout << "Programming "; }
   virtual void Pro2() { cout << "is "; }
   void Pro3() const { cout << "fun. "; }
   virtual void Con1() { cout << "You have to do it! "; }
};

class BasicProg : public Programming
{
public:
   virtual void Pro1() { cout << "Learn basics "; }
   virtual void Pro2() { cout << "was good "; }
   void Pro3() { cout << "but "; }
   virtual void Con1() { cout << "Bas Prog 4 "; }
};

class AdvProgramming : public Programming
{
public:
   virtual void Pro1() { cout << "Com Prog 1 "; }
   virtual void Pro2() { cout << "Com Prog 2 "; }
   void Pro3() { cout << "Com Prog 3 "; }
   virtual void Con1() { cout << "Com Prog 4 "; }
};

class Algo : public AdvProgramming
{
public:
   virtual void Pro1() { cout << "Algo 1 "; }
   virtual void Pro2() { cout << "Algo 2 "; }
   void Pro3() { cout << "Algo 3 "; }
   virtual void Con1() { cout << "Algo 4 "; }
};

class DS : public AdvProgramming
{
public:
   virtual void Pro1() { cout << "has been "; }
   virtual void Pro2()
   {
      cout << "the best. ";
      Programming::Con1();
   }
   void Pro3() { cout << "DS 3 "; }
   virtual void Con1() { cout << "DS 4 "; }
};

class PF : public BasicProg
{
public:
   virtual void Pro1() { cout << "PF "; }
   void Pro3() const { cout << "PF 3 "; }
   virtual void Con1() { cout << "PF 4 "; }
};

class OOP : public BasicProg
{
public:
```

```
    virtual void Pro1() { cout << "OOP "; }
    virtual void Pro2() { cout << "Reuse "; }
    void Pro3() { cout << "Polymorphism "; }
    virtual void Con1() { cout << "too many "; }
};

class Humanities : public Course
{

public:
    virtual void Pro1() { cout << "Important "; }
    virtual void Pro2() { cout << "to "; }
    virtual void Pro3() { cout << "learn "; }
    virtual void Con1() { cout << "None "; }
};

class Isl : public Humanities
{
};
class CommSkills : public Humanities
{
};

int main()
{
    Course* cp; AdvProgramming* app; Course co;
    PF p; OOP o; Algo a; DS d; CommSkills c; Isl i;
    Course& cr = p; Course& cr1(co); Humanities h;
    cp = &p;
    cp->Pro1();
    cr = o;
    cr.Pro2();
    cr.Pro3();
    cp = &o;
    cp->Pro1();
    app = &d;
    app->Pro1();
    app->Pro2();
    cr1 = h;
    cr1.Pro1();
    cr1.Pro2();
    cr1.Pro3();
}
```

Question – 51:

```
class Polygon
{
protected:
    int width, height;

public:
    void set_values(int a, int b)
    {
        width = a;
        height = b;
    }
    void print()
    {
```

```
        cout << width << " " << height << endl;
    }
};

class Rectangle : public Polygon
{
private:
    int length;

public:
    int set_values(int a, int b, int c)
    {
        Polygon::set_values(a, b);
        length = c;
    }
    void print()
    {
        cout << width << " " << height << " " << length << endl;
    }
};

void temp(Polygon& obj)
{
    obj.print();
}

int main()
{
    Rectangle rect;
    Polygon poly;
    rect.set_values(4, 5);
    poly.set_values(10, 20, 30);
    temp(rect);
    temp(poly);
    return 0;
}
```

Question – 52:

```
#include <iostream>
using namespace std;
class A
{
public:
    virtual const char* fun1(int x) { return "A"; }
    virtual const char* fun2(int x) { return "A"; }
};
class B : public A
{
public:
    virtual const char* fun1(short int x) { return "B"; }
    virtual const char* fun2(int x) const { return "B"; }
};
int main()
{
    B b;
    A& a(b);
    std::cout << a.fun1(1) << '\n';
    std::cout << a.fun2(2) << '\n';
```

```
    return 0;
}
```

Question – 53:

```
#include <iostream>
using namespace std;
class B;
class A
{
public:
    A() {}
    A(const A& copy) { cout << "f"; }
    A(const B& copy) { cout << "sf"; }
    virtual const char* fun1(int x) { return "A"; }
    virtual const char* fun2(int x) { return "A"; }
};

class B : public A
{
public:
    B() {}
    B(const B& copy) { cout << "fg"; }
    virtual const char* fun1(short int x) { return "B"; }
    virtual const char* fun2(int x) const { return "B"; }
};
int main()
{
    B b;
    A& a(b);
    std::cout << a.fun1(1) << '\n';
    std::cout << a.fun2(2) << '\n';
    return 0;
}
```

Question – 54:

```
#include <iostream>
using namespace std;
class B;
class A
{
public:
    A() {}
    A(const A& copy) { cout << "f"; }
    A(const B& copy) { cout << "sf"; }
    virtual const char* fun1(int x) { return "A"; }
    virtual const char* fun2(int x) { return "A"; }
};

class B : public A
{
    int a;
public:
    B() {}
    B(const B& copy) { cout << "fg"; }
    void doSomething()
    {
```

```
        a++;
    }
    virtual const char* fun1(short int x) { return "B"; }
    virtual const char* fun2(int x) const {
        this->doSomething();
        return "B";
    }
    virtual const char* fun2(int x) { return "B"; }
};
int main()
{
    B b;
    A& a(b);
    std::cout << a.fun1(1) << '\n';
    std::cout << a.fun2(2) << '\n';
}
```

Question – 55:

```
int i;
class LFC
{
    int x;
    int y;

public:
    LFC()
    {
        x = 0;
        cout << i << endl;
    }
    int getX()
    {
        return x;
    }
    LFC getY()
    {
        return *this;
    }
    ~LFC()
    {
        cout << i << endl;
        i = 10;
    }
};
int foo(LFC obj)
{
    i = obj.getY().getX();
    LFC ob;

    return i;
}
int main()
{
    LFC obj;
    cout << foo(obj) << endl;
    return 0;
}
```

Question – 56:

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

class A {

public:
   A(int ii = 0) : i(ii) {}
   A(int ii, int ss) : i(ii + ss) {}
   void show() { cout << "i = " << i << endl; }
   ~A() { cout << "Out A" << endl; }
   A magic(int ss) { this->i = ss * this->i; return *this; }

private:
   int i;
};

int main() {

   A a, s(10);
   s.magic(20);
   a.magic(10);
}
```

Question – 57:

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

class A {

public:
   A(int ii = 0) : i(ii) {}
   A(int ii, int ss) : i(ii + ss) {}
   void show() { cout << "i = " << i << endl; }
   ~A() { cout << "Out A" << i << endl; }
   A magic(int ss) { this->i = ss * this->i; return *this; }

private:
   int i;
};



int main() {

   A a, s(10);
   a = s.magic(20);

}
```

Question – 58:

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

class A {

public:
   A(int ii = 0) : i(ii) {}
   A(int ii, int ss) : i(ii + ss) {}
   void show() { cout << "i = " << i << endl; }
   ~A() { cout << "Out A" << i << endl; }
   A magic(int ss) { this->i = ss * this->i; return *this; }

private:
   int i;
};

int main() {

   A a(10, 5), s(10);
   a = s.magic(20).magic(10);

}
```

Question – 59:

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

class B {

public:
   B(int xx) : x(xx) { cout << "In B" << endl; }

   ~B() { cout << "Out B" << endl; }
   int getX() { return x; }
private:
   int x;
};

class A {

public:
   A(int ii = 0) : i(ii) {}
   A(int ii, int ss) : i(ii + ss) {}
   void show() { cout << "i = " << i << endl; }
   ~A() { cout << "Out A" << endl; }
   A(B abc) { i = abc.getX(); cout << "In A B" << endl; }

   A& operator=(B& b) { this->i = b.getX(); return *this; }
private:
   int i;
```

```
};


void g(A a)
{
   a.show();
}

int main() {
   B b(10);
   g(b);
   g({ 20, 30 });
   A a, s(10);
   a = b;
   s = a;

}
```

Question – 60:

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

class A {

public:
        A(int ii = 0) : i(ii) {}
        void show() { cout << "i = " << i << endl; }
private:
        int i;
};

class B {

public:
        B(int xx) : x(xx) {}
        operator A() const { return A(x); }
private:
        int x;
};

void g(A a)
{
        a.show();
}

int main() {
        B b(10);
        g(b);
        g(20);
}
```

Question – 61:

```
#include <iostream>
using namespace std;
class A
{

   int data[2];
   int arr[3];
   int ss;
public:
   A(int x, int y) : data{ x, y }, arr{ x + y, y - x, y % x }
   {
      ss = y / x;
   }

   A(int* ptr) : data{ *ptr, *(ptr + 1) }, arr{ 0 }
   {
      ss = *ptr;
   }


   void display() { cout << data[1] + ss + arr[2] << endl; }

   ~A() { cout << arr[0] - data[0] - ss << endl; }
};

int main()
{
   A a(22, 33);

   int* arr = (int*)&(a);
   arr += 3;
   cout << arr[-2] + arr[2] << endl;
   a = (arr - 2);
   a.display();

}
```

Question – 62:

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

class MyClass {
   int a, b;

public:
   MyClass(int i)
   {
      a = i;
      b = i;
   }
   void display()
   {
      cout << "a = " << a << " b = " << b << "\n";
   }
   ~MyClass() { cout << "His Class = " << a + b << endl; }
```

```
};

int main()
{
   MyClass object(10);
   object.display();

   object = 20;
   object.display();
}
```

Question – 63:

```
struct structure {
        int x;
        structure* ptr;
};
void print(structure* pointer)
{
        while (pointer != NULL)
        {
                cout << pointer->x << " -> ";
                pointer = pointer->ptr;
        }
        cout << "." << endl;
}
int main()
{
        structure three = { 10 }, two = { 30 }, one = { 20 }, * pointer = &one;
        three.ptr = &two; one.ptr = &three;
        print(pointer);
        structure four;
        four.x = 15; four.ptr = pointer;
        pointer = &four;
        print(pointer);
        return 0;
}
```

Question – 64:

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

class B {

public:
   B(int xx) : x(xx) { cout << "In B" << endl; }

   ~B() { cout << "Out B" << endl; }
   int getX() { return x; }
private:
   int x;
};

class A {
```

```
public:
   A(int ii = 0) : i(ii) {}
   A(int ii, int ss) : i(ii + ss) {}
   void show() { cout << "i = " << i << endl; }
   ~A() { cout << "Out A" << endl; }
   A(B abc) { i = abc.getX(); }

private:
   int i;
};

int main() {
   B b(10);

   A a, s(10);
   a = b;
   s = a;
}
```

Question - 65:

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

class ZooCage
{
   int cageNumber;
   ZooCage* link;

public:
   ZooCage(int n) : cageNumber(n), link(nullptr) {}

   int getCageNumber()
   {
      return this->cageNumber;
   }

   ZooCage*& getLink()
   {
      return this->link;
   }
};

ZooCage* start = nullptr;

void fun(ZooCage*& H, int num)
{
   if (H)
   {
      fun(H->getLink(), num);
      return;
   }
   H = new ZooCage(num);
}

void fun(ZooCage* H)
{
   if (H)
   {
```

```
        fun(H->getLink());
        cout << H->getCageNumber() << endl;
    }
}

int main()
{
    fun(start, 4);
    fun(start, 2);

    ZooCage* temp = new ZooCage(5);
    temp->getLink() = start->getLink()->getLink();
    start = temp;

    fun(start, 3);
    fun(start);
    return 0;
}
```

Question – 66:

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


class A {

public:
    A(int ii = 0) : i(ii) {}
    A(const A& abc)
    {
        this->i = abc.i; cout << "Out Of " << i - 1 << endl;
    }
    A magic(A abc) {

        this->i = i + 1;
        A bcd(this->i);
        return bcd;
    }
    ~A() { cout << "Out A " << i << endl; }

private:
    int i;
};

int main() {

    A b(3), a(4);
    a = b.magic(a).magic(b);
}
```

Question – 67:

```
#include<iostream>
```

```
using namespace std;


class A {

public:
   A(int ii = 0) : i(ii) {}
   A(const A& abc)
   {
      this->i = abc.i; cout << "Out Of " << i - 1 << endl;
   }
   A magic(A abc) {
      A bcd(2);
      return bcd;
   }
   ~A() { cout << "Out A " << i << endl; }

private:
   int i;
};

int main() {

   A b(3), a(4);
   b.magic(a);

}
```

Question - 68:

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

class A {

public:
   A(int ii = 0) : i(ii) {}
   A(const A& abc)
   {
      this->i = abc.i;
      cout << "Out Of " << i - 100 << endl;
   }
   ~A() { cout << "Out A" << i << endl; }
   A magic(int ss)
   {
      this->i = ss * this->i; return *this;
   }

private:
   int i;
};

int main() {

   A a(15), s(10);
   a = s.magic(20).magic(10).magic(3);
```

}

Question - 69:

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


class A {

public:
   A(int ii = 0) : i(ii), s(new int(i + 1)) {}
   A(const A& abc)
   {
      this->i = abc.i;
      this->s = new int(*(abc.s));
      cout << "Out Of " << i + *s << endl;
   }
   A magic(A abc) {
      A bcd(2);
      return abc;
   }

   ~A() { cout << "Out A " << i << endl; }

   void show() { cout << *s << endl; }
private:
   int i;
   int* s;
};

int main() {

   A b(3), a(4);
   b.magic(a).show();

}
```