1	`void fun(const int* ptr, const int N){` `  for(int i=0; i<N; i++, ptr++) {` `  *ptr = 5;` `  cout << *ptr;` `  }` `}` `int main(){` `  int arr[4] = {1,2,3,4};` `  fun(arr, 4);` `  return 0;` `}`	**[2 marks]** **Error: const int *ptr is a read only pointer, cannot assign 5** Output after removing const keyword **5555** We use the pointer to constant (const * datatype ptr) when we don't want the pointer to be able to change the value at the address it points
2	`char *findChar(char *str) {` `  char *ptr = str;` `  while (*ptr != 's')` `  ptr++;` `  return ptr;` `}` `int main(){` `  cout << findChar("mystring");` `  return 0;` `}`	**[2 marks]** **string** strings are char arrays, the findChar function returns a pointer pointing at 's' in the string. Since this is a string, cout displays all the characters in the string from 's' till the null character
3	`char *findChar(char *str) {` `  char *ptr = str;` `  while (*ptr != 's')` `  ptr++;` `  return ptr;` `}` `int main(){` `  cout << *findChar("mystring");` `  return 0;` `}`	**[2 marks]** **s** strings are char arrays, the findChar function returns a pointer pointing at 's' in the string. cout displays the *value* this pointer points at
4	`void print(const char* p){` `  for(int i = 0; i < strlen(p);){` `  cout<<p<<endl;` `  p++;` `  }` `}` `int main(){` `  char p[] ={'1','2','3','\0'};` `  print(p);` `  return 0;` `}`	**[3 marks]** **123** **23** **3** Strings are char arrays will null pointer at the end. Pointer to constants can point at constants and non-constants. strlen(p) return 3 (null char is not counted).
5	`void fun3(int&a){` `  a++;` `  cout<<a;` `}` `void fun2(int &a){` `  fun3(++a);` `  cout<<a;`	**[4 marks]** **4444** If there is a local and global variable with the same name, the local one is used by default.

	} void fun1(int &a){ fun2(++a); cout<<a; } int a=5; int main(){ int a = 1; fun1(a); cout<<a; return 0; }	
6	int g_One=1; void func(int* pInt){ pInt=&g_One; } void func2(int*& rpInt){ rpInt=&g_One; } int main(){ int nvar=2; int* pvar=&nvar; func(pvar); cout<<*pvar<<endl; func2(pvar); cout<<*pvar<<endl; return 0; }	**[2 marks]** **2** **1** In func() pointer parameter is passed by value, change made inside function only remains till function scope In func2() pointer parameter is passed by reference, any change made inside the function is retained outside the function also.
7	int main(){ char sstring[] = {'g', 'n', 'o', 'r', 'w','\0'}; char* chp = sstring; chp += 4; for(int i=0;i<5;i++){ cout <<*(chp-i); } return 0; }	**[3 marks]** **wrong** cout statement prints the value at which the pointer (chp) points. This will only print one character at a time.
8	int main(){ int data = 10; //address 200 int * const what; //address 300 cout<<what<<"\t"<<*what<<"\\"<<&what; return 0; }	**[2 marks]** **Error: a constant pointer, similar to constant variables, MUST be initialized when declared.** **int *const what = &data; //correction** **200 10\300** A constant pointer points at the same address during the entire program execution. Not to be confused with “pointer to constant”
9		**[2 marks]** **0**

	int main(){ int array[] = {1,2,3,4,5}; int *p = array; cout<<(p++ == array+1); return 0; }	Post-increment evaluated after == Before increment p stores the address of the first element in the array array+1 is the address of the second element in the array. Address of element 1 is not equal to the address of element 2
10	int main(){ const int x = 10; int *q = &x; int *const_ptr = q; cout << *const_ptr << endl; return 0; }	**[2 marks]** **Error: A simple int * cannot point at a constant variable. Only a pointer to constant can.** **const int *q=&x; //correction** **const int * const_ptr =q; //correction** **10** Both pointers q and const_ptr are pointing at the address of a constant variable, therefore both should be pointers to constant
11	int main(){ int arr[5]={1,5,9,11,15,19}; int i; for(i=0;i<5;i++) cout<<arr[i]/4*arr[i]/2<<"\t"; return 0; }	**[3 marks]** **Error: 6 elements are assigned to array of size 5** **int arr[6]={1,5,9,11,15,19}; //correction** **0 2 9 11 22**
12	int main(){ int list[10]={21,12,13,3,55,16}; int i; for(i=0;i<5;i++) { int temp=list[i]; list[i]=list[9-i]; list[9-i]=temp; } for(i=0;i<10;i++) cout<<list[i]<<"\t"; return 0; }	**[3 marks]** **0 0 0 0 16 55 3 13 12 21** When initializing an array using the initializer list, if the values in the initializer list are less than the array size, remaining elements are initialized as 0 for numeric arrays.
13	int main() { int i,j,Matrix[4][4]={1, 3, 6,2,5, 9,1, 7,8 , 4, 5 ,3,4,5,6,9}; for(i=0,j=N-1 ; i<N ; i++,j--) { if (Matrix[i][j]%4==0)	**[2 marks]** **Error: N is not declared in this scope** **int N = 4; //correction** **1 0 5 3 5 3**

	cout<<Matrix[i][j]+1<<" "; cout<<Matrix[i][j]-1<<" "; } return 0; }	
14	int main() { int i,j,Matrix[3][3]={1,2,3,4,5,6,7,8,9}; for(int i=0;i<3;i++) { for(int j=0;j<3;j++) { if(i==j) cout<<Matrix[i][j]<<" "; } } }	**[2 marks]** **1 5 9**
15	int main(){ int i = 50,j = 1, x=0 ; do{ i= ++j; x++; }while(x<5); cout<<i<<" "<<j; }	**[2 marks]** **6 6** Prefix increment evaluated before the assignment
16	int main(){ for(int i=0;;){ i++; cout<<i<<" "; if(i==3) break; } }	**[2 marks]** **1 2 3** Stopping condition and update can be skipped in the for loop header
17	int main(){ int something = 1; for(int i = n ; i>=0; i--){ something = something * i; if(i==2) continue; if(i<3) break;	**[2 marks]** **Error: n isn't declared** **int n = 2; //correction** **2**

	`    }` `    cout<<something;` `}`	
18	`int main()` `{` `    int i=0, j=1;` `    while(i<5)` `    {` `        while(j<5){` `            cout<<"* ";` `            j++;` `        }` `        cout<<endl;` `        i++; j=i;` `    }` `    return 0;` `}`	**[2 marks]** `* * * *` `* * * *` `* * *` `* *` `*`
19	`int main()` `{` `    int i = 0, j=1, c=0;` `    while(j - ++i) {` `        c++;` `    }` `    cout<<"Executed "<<c<<" times\n";` `    return 0;` `}`	**[2 marks]** **`Executed 0 times`** Prefix executed before anything else in the statement 1 – 1 = 0 if the loop condition is 0 (false) the loop does not execute
20	`int main()` `{` `    switch(~(12|25))` `    {` `        case 0:` `            cout<<"Programing ";` `        case 1:` `            cout<<"Fundamentals!";` `            break;` `        case -12:` `        case 29:` `            cout<<"is";` `            break;` `        case -29:` `            cout<<"fun";` `            break;` `        default:` `            cout<<"None of the case is true";` `    }` `    return 0;` `}`	**None of the case is true** Bitwise OR of 12 and 25 = 29 Then bitwise NOT of 29 = -30
21	`int calculation(int n) {` `    if (n > 1) {` `        return n * (n - 1);` `    } else {`	**`result = 20`**

	<pre>return 1; } } int main() { int n, result; n=5; result = calculation(n); cout << "result = " << result; return 0; }</pre>	
22	<pre>int main() { const int UPPER = 7, LOWER = 6; int num1, num2, num3 = 12, num4 = 3; num1 = num3 < num4 ? LOWER: UPPER; num2 = num4 > UPPER ? num3 : LOWER; cout << num1 << " " << num2 << endl; return 0; }</pre>	**7 6**
23	<pre>int main() { int limit = 10; cout<<((limit++) && (++limit - 12)) ; }</pre>	**0** The logical && has two expressions, one on each side. First left one is evaluated and then right one. The left expression only has one increment. Limit becomes 11. Right expression has prefix so limit becomes 12 and then 12 is subtracted from it. 12-12 = 0 Even if one expression is false the whole AND condition is false. Note: adding brackets does not change the order in which the postfix or prefix are evaluated. E.g. a++ + b; and (a++) + b; work the **SAME** way.
24	<pre>#include <iostream> using namespace std; int main() {</pre>	**Error: break statement can only be placed in a loop or switch** **Remove break statement**

	`int n=10;` `{` `n=20;` `break;` `n=30;` `}` `cout<<n;` `return 0;` `}`	**30**
25	`#include <iostream>` `using namespace std;` `void test(int a);` `int main(){` `test(10);` `}` `void test(int b){` `a = 20;` `b = 30;` `cout<<"a + b = "<< a * b;` `}`	**Error: Variable a is not defined in test function.** **Not an error but prototype should not have variable name** **int a=20; //correction** **600**
26	`#include <iostream>` `using namespace std;` `int do_something(int);` `int main(){` `cout<<do_something(5);` `}` `int do_something(int n){` `int something = 1;` `for(int i = n ; i>=0; i--){` `something = something * i;` `if(i==2)` `continue;` `if(i<3)` `break;` `}` `cout<< something;` `exit(0);` `return 1;` `}`	**120**

27	`#include <iostream>` `using namespace std;` `int main(){` `int a, b = 0;` `if(a=a+b)` `a = 2 * ++b + a++;` `switch(a){` `case 2:` `b = 2 * a;` `default:` `b = (true ? (a > 0 ? 10 : 20 ) : 30);` `break;` `case 0:` `b = (a > 0 ? 1 : 2 );` `case 3:` `b = a + 1;` `break;` `}` `cout<<a<<" "<<b;` `}`	**0 1** Assignment statement in if assigns the value 0 to a. If statement does not execute if the condition is 0 (false). Value of a is 0, case 0 is executed, but since there is no break after it, case 3 is also executed.
28	`#include <iostream>` `using namespace std;` `int main(){` `int a = 2, b = 2, c =3, d =4;` `a = a > b ? b : c > d ? c : d;` `cout << a << endl;` `}`	**4**
29	`#include <iostream>` `using namespace std;` `int main()` `{` `int testVal = 0;` `while (testVal++ < 10)` `{` `if (testVal == 4)` `continue;` `testVal = testVal+1;` `cout << testVal << " ";` `}` `return 0;` `}`	**[2 marks]** **2 4 6 8 10** Post-increment, happens after comparison
30	`#include <iostream>` `using namespace std;` `int func (int);` `int main()` `{` `int x = 2, y = 3, z = 4;` `cout << "values of x, y, z before function calls` `are : " << x << " , " << y << " , " << z<<endl;`	**[3 marks]** **Error: 'i' was not declared in this scope.** **for(int I=1, I<=x; I++) //correction** **values of x, y, z before function calls are : 2 , 3 , 4**

	int x_value = func (x); int y_value = func (y); int z_value = func (z); cout << “values of x, y, z after function calls are : “ << x_value << “ , “ << y_value << “ , “ << z_value<<endl; return 0; } int func (int x = 6) { int temp=1; for(int I = 1; I <= x; i++) temp *= I; x = temp; return x; }	**values of x, y, z after function calls are : 2 , 6 , 24**
31	#include <iostream> using namespace std; int main() { int x = 9, y = 11; if (x < 10); cout << "@@@" << endl; if (y > 10) if (y> x); cout << "!!!" << endl; if (x==y) cout << "***" << endl; else cout << "###" << endl; cout << "$$$" << endl; return 0; }	**[2 marks]** **@@@** **!!!** **###** **$$$**
32	int main(){ double value = 92.8762; double *a, b; a = &value; b = a; cout<<*b; return 0; }	**Error: cannot save address in a double variable. Address can only be saved in a pointer.** **double *a,*b; //correction** **92.8762** Pointer must be of the same type as the variable it points to

Q1. Write an operator overloading functions for the following code. Assume getters and setters are already available in each class, so you don't have to rewrite them.

```
Class A
{
int salary;
int awards;

public:
A (): salary (5000), awards (0) { }

//Write prototype if any

_____________________________________________________

_____________________________________________________
};

Class B
{
int salary;
int awards;

public:
B (): salary (12000), awards (2) { }

//Write prototype if any

_____________________________________________________

_____________________________________________________
};

void main ()
{
  A emp1;
  B m1;
  int total = emp1 * m1;      //should multiply only awards
  A emp2 = --emp1;
  B m2 = m1++;
}
```

Solution is in red color:

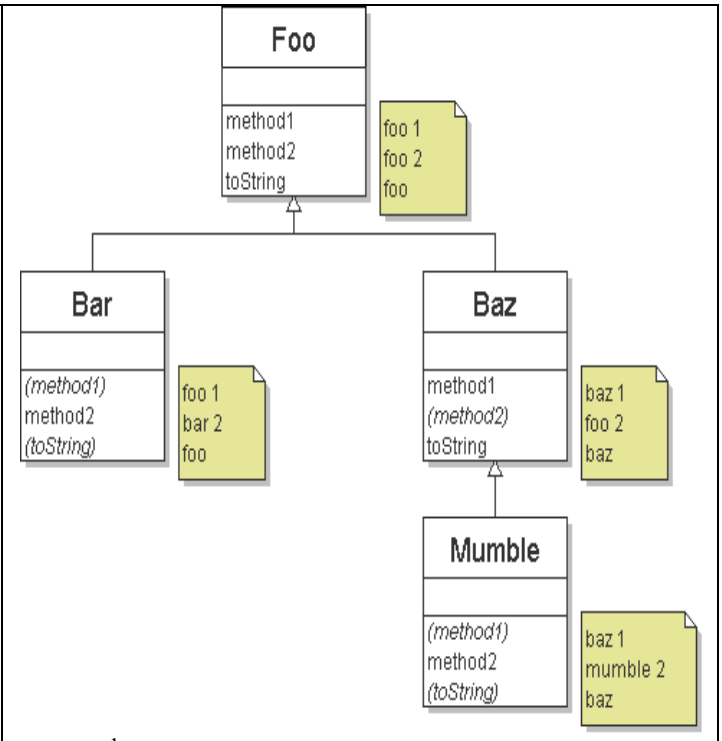
```
#include <iostream>
using namespace std;

 class B
{
int salary;
int awards;

public:
B (): salary (12000), awards (2) { }
B operator++ (int); //postfix

int getsalary(){
    return salary;

}

int getawards(){
    return awards;

}
};

class A
{
int salary;
int awards;

public:
A (): salary (5000), awards (0) { }
int operator* (B&);
A operator-- ();   //prefix
```

```
int getsalary(){
    return salary;

}

int getawards(){
    return awards;

}
};


int A::operator* (B& man){
    int total;
    total = this -> awards * man.getawards();
    return total;

}

A A::operator-- (){    //prefix
    A emp;
    salary--;
    awards--;
    emp.salary = this->salary;
    emp.awards= this -> awards;


    return emp;
}

B B::operator++ (int){    //postfix
    B m;

    m.salary = this->salary;
```

```
    m.awards= this -> awards;
    salary++;
    awards++;


    return m;
}


int main ()
{
  A emp1;
  B m1;
  int total = emp1 * m1;     //should multiply only awards
  A emp2 = --emp1;
  B m2 = m1++;

  // These cout statements are just to check the solution. You were not required to write them.

  cout << total << endl;
  cout << emp2.getsalary() << endl;
  cout << emp2.getawards() << endl;
  cout << m1.getsalary() << endl;
  cout << m1.getawards() << endl;
  cout << m2.getsalary() << endl;
  cout << m2.getawards() << endl;

      return 0;
}
```

Practice Problems: Inheritance & Polymorphism

```
public class Foo {
   public void method1() {
      System.out.println("foo 1");
   }
   public void method2() {
      System.out.println("foo 2");
   }
   public String toString() {
      return "foo";
   }
}
public class Bar extends Foo {
   public void method2() {
      System.out.println("bar 2");
   }
}

public class Baz extends Foo {
   public void method1() {
      System.out.println("baz 1");
   }
   public String toString() {
      return "baz";
   }
}
public class Mumble extends Baz {
   public void method2() {
      System.out.println("mumble 2");
   }
}

public class Polymorphism{
public static void main(String [] args){
Foo[] pity = { new Baz(), new Bar(),
            new Mumble(), new Foo() };
for (int i = 0; i < pity.length; i++) {
   System.out.println(pity[i]);
   pity[i].method1();
   pity[i].method2();
   System.out.println();
}
}
```




```
baz
baz 1
foo 2

foo
foo 1
bar 2

baz
baz 1
mumble 2

foo
foo 1
foo 2
```

*( ) method is inherited. Otherwise, method is overridden.

1. **Tracing programs: The above is the program demonstrated in class. Now, what gets printed to the screen when we execute the following classes on the left?**

`` public class A { public int x = 1; public void setX(int a){ x=a; } } public class B extends A { public int getB(){ setX(2); return x;} } public class C { public static void main(String [] args){ A a = new A(); B b = new B(); System.out.println(a.x); System.out.println(b.getB()); } } ```	Result: 1 2 Public instance variable and instance method can be inherited and accessed by subclass (without overriding)
`` public class A { private int x = 1; protected void setX(int a){ x=a; } protected int getX(){ return x;} } public class B extends A { public int getB(){ setX(2); //return x; It does not work because private modifier, so return getX(); } } public class C { public static void main(String [] args){ A a = new A(); B b = new B(); System.out.println(a.getX());//a.x is not allowed, private! System.out.println(b.getB()); } } ```	Result: 1 2 Private instance variable and private instance methods can be inherited but not accessible to subclass! Protected instance variable and protected instance methods can be inherited and accessible to subclass,
`` public class A { protected int x = 1; protected void setX(int a){x=a;} protected int getX(){return x;} } public class B extends A { public int getB(){ setX(2); return x; } } public class C { public static void main(String [] args){ A a = new A(); B b = new B(); System.out.println(a.getX()); System.out.println(b.x); //b.x is protected, then inherited. System.out.println(b.getB()); } } ```	Result 1 1 2 *The difference of B's x is not variable shadowing. It's the expected execution of value resetting (setX(2)).

```
public class A {
protected int x = 1;
protected void setX(int a){
x=a;
}
protected int getX(){
return x;}
}
public class B extends A {
protected int x = 3;
public int getX(){
return x; }
public int getB(){
return x;
}
}
public class C {
public static void main(String [] args){
A a = new A();
B b = new B();
System.out.println(a.getX());
System.out.println(b.getB());//subclass method access own attrib
System.out.println(b.getX());//overriding method, accessing sub
System.out.println(a.x); //protected
System.out.println(b.x); //overriding attribute!
}
}
```

Resutls:
1
3
3
1
3

Do you know which getX of b is called, A's or its own? If you cannot ensure your answer right, please see the comment in the below.

```
public class A {
protected int x = 1;
protected void setX(int a){
x=a;
}
protected int getX(){
return x;}
}
public class B extends A {
protected int x = 3;
public int getX(){
return x; }
public int getB(){
return x;
}
}
public class C {
public static void main(String [] args){
A a = new A();
A b = new B();   //polymorphism, making shadowing possible!
System.out.println(a.getX());
System.out.println(b.getX());//override, access subclass attri.
//System.out.println(b.getB()); not able to load subclass method!
System.out.println(a.x);
System.out.println(b.x);  //variable shadowing!
}
}
```

Results:
1
3
1
1

Subclass variable can be accessed by method, the direct access (without using method) will reach the overridden value from superclass!

b.getB is not permitted because it is out A's signature. b.getX is allowed because it is overridden!

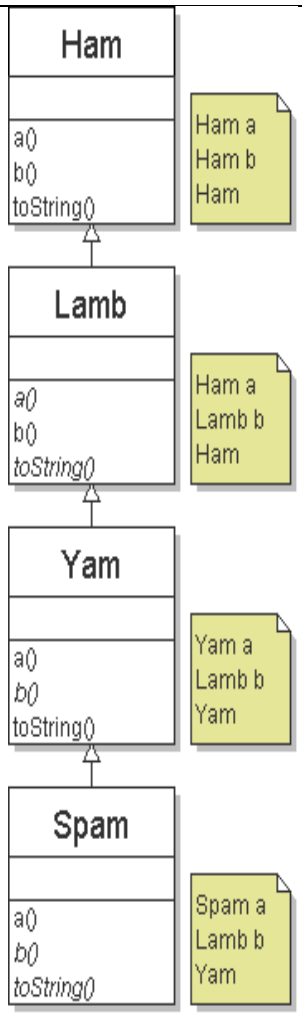
// For your development:
//1) Is it good to block the use of b.getB()?
// ANS>: Good, because methods can be in template. In the security control, no leakage!
//2) Is it good to have the direct access of attribute such as b.x?
// ANS>: Better not, if it is not in your control. See how complicate it is in this program.

```
public class A {
protected int x = 1;
protected void setX(int a){
x=a;
}
protected int getX(){
return x;}
}
public class B extends A {
protected int x = 3;
public int getX(){
setX(2); // call superclass method to set superclass attrib
return x; } //but return attrib of subclass
public int getB(){
return x;
}
}
public class C {
public static void main(String [] args){
A a = new A();
A b = new B();
System.out.println(a.getX());
System.out.println(b.getX());
System.out.println(a.x);
System.out.println(b.x);
}
}
```

Results:
1
3
1
2

b.x is set to 2 because a superclass method is called to change the value of shadowed value.

```
public class Ham {
    public void a() {
        System.out.println("Ham a");
    }
    public void b() {
        System.out.println("Ham b");
    }
    public String toString() {
        return "Ham";
    }
}

public class Lamb extends Ham {
    public void b() {
        System.out.println("Lamb b");
    }
}

public class Yam extends Lamb {
    public void a() {
        System.out.println("Yam a");
    }
    public String toString() {
        return "Yam";
    }
}

public class Spam extends Yam {
    public void a() {
        System.out.println("Spam a");
    }
}



public class Polymorphism2 {
public static void main (String [] args){
Ham[] food = { new Spam(), new Yam(),
               new Ham(), new Lamb() };
for (int i = 0; i < food.length; i++) {
    System.out.println(food[i]);
    food[i].a();
    food[i].b();
    System.out.println();
    }
}
}
```



Yam
Spam a
Lamb b

Yam
Yam a
Lamb b

Ham
Ham a
Ham b

Ham
Ham a
Lamb b

```
public class Ham {
        int a = 0;
             int b = 1;
    public void a() {
        System.out.println("Ham " + a);
    }
    public void b() {
        System.out.println("Ham " + b);
    }
    public String toString() {
        return "Ham " + a + " " + b;
    }
}

public class Spam extends Ham {
        int a = 2;
    public void a() {
        System.out.println("Spam " +a);
    }
}

public class Yam extends Spam {
        int b = 3;
    public void a() {
        System.out.println("Yam " + a);
    }
    public void b() {
        System.out.println("Yam " + b);
    }
}

public class Polymorphism3 {
public static void main (String [] args){
Ham[] food = { new Spam(), new Yam(),
               new Ham()};
for (int i = 0; i < food.length; i++) {
    System.out.println(food[i]);
    food[i].a();
    food[i].b();

System.out.println(food[i].a);

System.out.println(food[i].b);
    System.out.println();
}
}
}
```

```
Ham 0 1
Spam 2
Ham1
0
1

Ham 0 1
Yam 2
Yam 3
0
1

Ham 0 1
Ham0
Ham1
0
1
```

```
public class A
{
  private String x = "Ax";
  protected String y = "Ay";
  public String z = "Az";

  public String toString() {
    return x + y + z;
  }

public static void main(
String [] args)
{
A a = new A();
System.out.println(a);
}
}

public class B extends A
{
  private String x = "Bx";
public String z = "Bz";

public String toString() {
    return x + y + z;
  }

  public static void main(
String [] args)
  {
    B b = new B();
    System.out.println(b);
  }
}
```

```
public class C extends A
{
private String x = "Cx";

  public static void main(
                        String [] args)
  {
    C c = new C();
    System.out.println(c.x);
    System.out.println(c);
  }
}

public class D extends C
{
  private String x = "Dx";
  public String z = "Dz";

public static void main(
                        String [] args)
  {
D d = new D();
System.out.println(d.x);
    System.out.println(d.y);
    System.out.println(d.z);
    System.out.println(d);

C c = new D();
// Error: System.out.println(c.x);
    System.out.println(c.y);
    System.out.println(c.z);
    System.out.println(c);
  }
}
```

When A is executed, it displays:

```
AxAyAz
```

The println statement implicitly calls a.toString(), which creates a string containing the concatenation of the variables x, y, and z. Once this concatenated String ("AxAyAz") is returned, it gets printed to the screen.

When B is executed, it displays:

```
BxAyBz
```

The println statement implicitly calls b.toString(), which refers to the overriding toString() method defined in subclass B. This toString method says to concatenate x+y+z (like the one defined in class A), but now it is referring to variables x, y, and z in the B class. Two of those variables are defined locally – x and z. These variables hide (or "shadow") the x and z variables defined in class A. The third variable, y, is inherited from A. As a result, the concatenation in the toString method produces a String that looks like "BxAyBz", which then gets printed.

When C is executed, it displays:

```
Cx
AxAyAz
```

When the first println statement executes, it refers directly to c.x, which is defined locally. So that prints out "Cx".
When the second println statement executes, it refers to C's inherited toString method. The inherited toString method is defined in class A, which returns the concatenation of x, y, and z from class A. So that returns "AxAyAz", which gets printed.

When D is executed, it displays:

```
Dx
Ay
Dz
DxAyDz
Ay
Az
DxAyDz
```

The first 3 lines print the values of x, y, and z inside of d. Since x and z are defined inside of class D, those values get printed out. y is inherited from class A, so "Ay" gets printed out.

The next line prints the return value of D's toString method. The toString method is defined locally to override the inherited one (unlike in the example for class C, where the toString method is inherited instead of overridden). Because the toString method is overridden, when it refers to x, y, and z, it refers to the variables inside of class D. So this returns "DxAyDz". Compare this to the inherited toString method in class C, which returns "AxAyAz".

The next two lines use a variable with static type C to refer to an object of dynamic type D. Notice that it is an error to try to print (or refer to) c.x. That's because 'x' is private in class C, and this code is written inside class D. Also notice that c.z is "Az", whereas d.z is "D.z". For fields, Java uses the value of the static type's field (in this case, the value of z from class C, which is inherited from class A and has value "Az").

The last line prints the value of c.toString(). Java uses the value of a the *static* type's *field*, but the *dynamic* type's *methods*. Variable c has dynamic type D, because it refers to an object of type D. So Java uses the toString method defined in class D, which returns the values of x, y, and z within class D (or "DxAyDz"). Notice the difference between how fields get handled, and how methods get handled. The field c.z refers to the field defined in class C (which is inherited from class A). The method c.toString() refers to the method defined in class D, not class C.

I have still not figured out any reason why Java does shadow things this way for fields. It's very confusing, and it can lead to very hard-to-fix bugs. In general, it is HIGHLY RECOMMENDED that you AVOID defining fields with the same name as a superclass's field. Sometimes though (like when you're extending a superclass from the Java API), you may not know what the superclass's fields are called, and in that case, you just have to guess.

2. Program Development

Here's a problem from a previous Final Exam.

Consider the following skeleton for a Robot class, which has private fields for storing the location of aRobot object, its name, and the direction it’s facing (North for a direction parallel to

the positive y axis,South for the negative y axis, East for the positive x axis, or West for the negative x axis). It also hasstub methods for constructing a Robot object, changing the direction, and moving the location of therobot in the direction it's facing.

```
public class Robot
{
      private String name;

      private char direction; //'N','S','E', or 'W'

      privateintxLoc, yLoc; // the (x, y) location of the robot

      // Initialize name, direction, and (x, y) location
      public Robot(String name, char dir, int x, int y) { ... }

      public String toString()
      {
            return name + " is standing at (" + x + "," + y + ") and facing"
            + direction);
      }

      // turn 90 degrees clockwise, e.g. 'N' changes to 'E', 'E' to 'S', ...
      public void turnClockwise() { ... }

      // turn 90 degrees counterclockwise, e.g. 'N' to 'W', 'W' to 'S', ...
      public void turnCounterClockwise() { ... }

      // move numSteps in direction you are facing,
      // e.g. if 'N' 3 steps, then y increases 3
      public void takeSteps(intnumSteps) { ... }
}
```

(a)Assuming the class above is completed correctly, what does the following program display on the screen:

```
public static void main(String args[])
{
      Robot robby = new Robot("Robby", 'N', 10, 12);
      for (inti = 0; i< 5; i++)
      {
            if (i % 2 == 0)
            {
                  robby.turnClockwise();
            }
            else
            {
                  robby.turnCounterClockwise();
            }

            robby.takeSteps(3);
            System.out.println(robby);
      }
}
```

Displayed on screen:

```
Robby is standing at (13, 12) and facing E
Robby is standing at (13, 15) and facing N
Robby is standing at (16, 15) and facing E
Robby is standing at (16, 18) and facing N
Robby is standing at (19, 18) and facing E
```

(b) Complete the constructor, the `turnClockwise`method, and the `takeSteps`method. Make sure your constructor validates its input. You do not need to define `turnCounterClockwise`.

```
public Robot(String name, char dir, int x, int y)
{
this.name = name;
this.direction = dir;
  this.xLoc = x;
  this.yLoc = y;
}

public void turnClockise()
{
  if(direction=='N') { direction = 'E'; }
  else if(direction=='E') { direction = 'S'; }
  else if(direction=='S') { direction = 'W'; }
  else { direction = 'N'; }
}

public void takeSteps(int numSteps)
{
if(direction=='N') { yLoc += numSteps; }
  else if(direction=='E') { xLoc += numSteps; }
  else if(direction=='S') { yLoc -= numSteps; }
  else { xLoc -= numSteps; }
}
```

(c) Write Java code to create an array of 5 robots. Use a for loop to fill in the array so that the n-th robot is named “robot n”, and it starts off life facing east at location (n, n).

```
Robot [] robots = new Robot[5];
for(int i=0; i<robots.length; i++)
{
  robots[i] = new Robot("robot " + i, 'E', i, i);
}
```

Here's another problem from a previous Final Exam. This one is an inheritance/polymorphism question.

```
classSuperClass
{
     protectedint x = 0;

     publicSuperClass(int x)
     {
          this.x = x;
     }

     private void increment() { x++; }

     protected final void add(int y)
     {
          x += y;
     }

     public void display()
     {
          System.out.println(x);
     }
}

public class SubClassextends SuperClass
{
     publicSubClass(int x)
     {
          super(x);
     }

     public void display()
     {
          add(2);
          super.display();
     }

     public static void main(String [] args)
     {
          SuperClasssc = new SuperClass(3);
          sc.display();

          sc = new SubClass(3);
          sc.display();
     }
}
```

(a)List the name of all methods that subclasses of SuperClass inherit.

subclasses inherit all methods: the constructor, increment, add, and display. If you want, you could also list all of the methods that SuperClass implicitly inherits from the Object class (eg, equals, toString, etc.), but that's not required.

(b)List the name of all methods that are visible in subclasses of SuperClass (in other words, methodsthat can be called directly).

add can be called directly just be using the name add(). the constructor SuperClass can be called by using the super() constructor. the display() method from SuperClass is overridden by the display() method in the SubClass, but it can still be called by writing super.display(). In summary, any method that has public or protected access in the superclass can be called directly by the subclass.

(c)List the name of all methods that may NOT be overridden by any subclasses of SuperClass.

*methods that are declared to be **final** in the superclass may not be overridden. So the add() method may not be overridden.*

(d)What gets displayed on the screen when SubClass is executed?

displayed on screen:

```
3
5
```

Write the output of the following programs (if any). If there is an error in the program, mention the error and move on.

Tip: Use python tutor (*https://pythontutor.com/cpp.html#mode=edit*) for line-by-line execution of programs for a better understanding, however, first try to solve by yourself.

Code	Output
void foo(int* arr1, const int size, int val, int* pos) { if(*pos == size - 1) *arr1 = val; else { *arr1 = val; ++*pos; foo(arr1 + 1, size, val, pos); } } int main() { const int size = 5; int arr[size] = {10, 20, 33, 0, 1}; int pos = 0; foo(arr, size, 10, &pos); for(int i = 0; i < size; ++i) cout<<arr[i]<<" "; return 0; }	10 10 10 10 10
void make2(int *arr, int cols) { arr = new int[cols]; } void make1(int **arr, int rows, int cols) { arr = new int*[rows]; make2(*arr, cols); } void make(int ***arr, int pages, int rows, int cols) { arr = new int**[pages]; make1(*arr, rows, cols); } int main() { int*** arr = NULL; make(arr, 4, 4, 4);	Segmentation fault in the inner most loop in the main. Reason, memory is being allocated within the function, however, pointers are being passed by value, thus the value of pointer in the main as well in the functions didn't update after returning back.

for(int i = 0; i < 4; ++i) { for(int j = 0; j < 4; ++j) { for(int k = 0; k < 4; ++k) arr[i][j][k] = i + j + k; } } return 0; }	
int main() { int num[5]= {1,2,3,4,5}; int* p; p = num; *p = 20; p = &num[1]; *(++p) = 30; p = num + 4; *p = 30; p = num; *(p + 3) = 40; for (int i = 1; i < 5; i++) cout << num[i] << " "; return 0; }	2 30 40 30
int main() { char name[5][10] = { "Pakistan", "China", "Turkiye", "Korea", "Japan"}; char* ptr1 = name[0]; cout<<ptr1<<endl; cout<<*ptr1<<endl; ptr1 = name[3]; cout<<ptr1<<endl; cout<<*(ptr1 + 1)<<endl; cout<<ptr1+2<<endl; ptr1 = name[1]; cout<<*(ptr1 + 10)<<endl; ptr1 = name[2]; cout<<ptr1-10<<endl; return 0; }	Pakistan P Korea o rea T China

int main() { char* alpha, beta; beta = new char[5]; return 0; }	Syntax error beta is of type char. It cannot allocate memory.
int main() { int alpha = 100, beta = 200; int *p = &alpha, *q = β p = q; cout<<p<<endl; cout<<*p<<endl; return 0; }	Address of beta variable 200
int main() { int a = 5, b = 10, c = 15; int *arr[] = {&a, &b, &c}; cout << arr[1]; return 0; }	Address of b
int main() { int i, j, var = 'A'; for (i = 3; i >= 1; i--) { for (j = 0; j < i; j++) { if(((i+var + j))%4==0) continue; cout<<char (i+var + j); } cout<<endl; } return 0; }	EF C B
int main() { char arr[20]; int i; for (i = 0; i < 10; i++) *(arr + i) = 65 + i; *(arr + i) = '\0';	ABCDEFGHIJ

cout << arr; return 0; }	
int main() { int*** arr = new int**[5]; for(int i = 0; i < 5; ++i) { arr[i] = new int*[5]; for(int j = 0; j < 5; ++j) arr[i][j] = new int[5] {1, 2, 3, 4, 5}; } for(int i = 0; i < 5; ++i) { for(int j = 0; j < 5; ++j) { for(int k = 0; k < 5; ++k) cout<<arr[i][j][k]<<" "; cout<<endl; } } delete arr; arr = NULL; return 0; }	Logical Error Dynamic Memory deletion done incorrectly resulting in memory leak.
int main() { int a[2][4] = {3, 6, 9, 12, 15, 18, 21, 24}; cout << *(a[1] + 2) << * (*(a + 1) + 2); return 0; }	2121
int main() { int* scores; scores = new int[4]{45, 65, 77, 67}; if(scores) cout<<++*scores++<<" "<<++*scores++<<" "<<++*scores++<<" "<<++*scores++;	46 66 78 68 Segmentation Fault Deleting unallocated memory

```
  if(scores)
    delete scores;
  scores = NULL;

  return 0;
}
```

Write the output of the following programs (if any). If there is an error in the program, correct the code and then write the output.

Tip: Use python tutor (https://pythontutor.com/visualize.html#mode=edit) for line by line execution of programs for a better understanding, first try to solve by yourself.

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; }	
const char* c[] = {"PF", "Exam" ,"PFMID-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; }	
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; }	
void f1(int*, int); void f2(int*, int);	

```
int main()
{
	int a;
	int b;
	a = 3;
	b = 5;
	f1(&a, b);
	f2(&a, b);
	cout << a << "," << b << ",";
	cout << a << "," << b;
}

void f1(int* p, int q)
{
	int tmp;
	tmp = *p;
	*p = q;
	q = tmp;
}

void f2(int* p, int q)
{
	int tmp;
	tmp = *p;
	*p = q;
	q = tmp;
}
```

```
int fun2(char* a, char* b)
{
	for (; *a == *b; a++, b++)
		if (*a == '\0')
			return 0;
	return *a - *b;
}


int main() {
	char a[10] = "date", b[10] = "data";
	cout << fun2(a, b) << endl;
}
```

```
void main()
{
	void* vp;
	char ch = 'g', * cp = "goofy";
	int j = 20;
	vp = &ch;
	cout << *(char*)vp;
	vp = &j;
	cout << *(int*)vp;
	vp = cp;
	cout << (char*)vp + 3 << endl;

}
```

int main() { char* ptr; char myString[] = "programing I"; ptr = myString; ptr += 5; cout << ptr; }	
int main() { int x = 20; int& y = x; int* p = &x; x = x + 20; y = y + 50; cout << *p << " " << y; }	
int main() { int data = 10; int const* what; what = &data; cout << what << "\t" << *what << "\\" << &what; return 0; }	
int main() { int array[] = { 1,2,3,4,5 }; int* p = array; cout << (p + (10 - 5) / 2 == array + 1); return 0; }	
int g_One = 1; void func(int* pInt) { pInt = &g_One; } void func2(int*& rpInt) { rpInt = &g_One; } int main() { int nvar = 2; int* pvar = &nvar; func(pvar); cout << *pvar << endl; func2(pvar); cout << *pvar << endl; return 0; }	

```
int main(){
	char* s[4] = { "black", "white",
		"yellow", "violet" };

	cout << (*(s + 1) + 2) << endl;
	cout << *(*(s + 2) + 3);
	return 0;
}
```

```
void f(int* p, int* q, int* k)
{
	p = q;
	f = p;
	q = f;
	*p = 2, * q = *f + 3; *f = *f + 1;
}
int i = 0, j = 1, f = 6;
int main()
{
	f(&i, &j, &f);
	cout << i << f << j;
	return 0;
}
```

```
void fun(int* p, int* s) {
	s = p;
	*s = 10;
	int x = 5;
	s = &x;
	return;
}
int main() {
	int x = 5;
	int* p = &x;
	int* s;
	s = &x;
	fun(p, s);
	cout << "x = " << x << " *p=" <<
		*p << " *s=" << *s;
	return 0;
}
```

```
const int s = 3;
int* listMystery(int list[][::s]) {
	int i = 1, k = 0;
	int* n = new int[::s];
	for (int i = 0; i < ::s; ++i)
		n[i] = 0;
	while (i < ::s)
	{
		int j = ::s - 1;
		while (j >= i)
		{
			n[k++] = list[j][i]
				* list[i][j];
			j = j - 1;
		}
		i = i + 1;
	}
	return n;
}
void displayMystery(int* arr) {
```

```
        cout << "[ ";
        for (int i = 0; i & lt; ::s; ++i)
                cout << arr[i] << (i != (::s - 1)
                        ? " , " : "  ");
        cout << " ] " << endl;
}

int main() {
        int L[][::s] = { {8, 9, 4}, {2, 3, 4},
                {7, 6, 1} };
        int* ptr = listMystery(L);
        displayMystery(ptr);
        delete[] ptr;
        return 0;
}
```

```
void function(char** ptr)
{
        char* ptr1;
        ptr1 = (ptr += sizeof(int))[-2];
        cout << ptr1 << endl;
}

int main()
{
        char* arr[] = { "ant", "bat", "cat",
                "dog", "egg", "fly" };
        function(arr);
        return 0;
}
```

```
int main() {

        int number1 = 88, number2 = 22;
        int* pNumber1 = &number1;
        *pNumber1 = 99;

        cout << *pNumber1 << endl;
        cout << &number1 << endl;
        cout << pNumber1 << endl;
        cout << &pNumber1 << endl;

        pNumber1 = &number2;
        int& refNumber1 = number1;
        refNumber1 = 11;

        cout << refNumber1 << endl;
        cout << &number1 << endl;
        cout << &refNumber1 << endl;

        refNumber1 = number2;
        number2++;

        cout << refNumber1 << endl;
        cout << number1 << endl;
        cout << number2 << endl;
        return 0;
}
```

```
int f(int x, int* py, int** ppz)
```

```
{
        int y, z;
        **ppz += 1;
        z = **ppz;
        *py += 2;
        y = *py;
        x += 3;
        return x + y + z;
}

int main()
{
        int c, * b, ** a;
        c = 4;
        b = &c;
        a = &b;
        cout << f(c, b, a);
        return 0;
}
```

```
int main()
{
   const int* p;
   const int a = 2;
   p = &a;
   *p = 7;
   cout << *p;
}
```

```
int main()
{
   const int a = 2;
   const int* p = &a;
   int b = 3;
   p = &b;
   cout << *p;
}
```

```
int main()
{
        int a[3] = { 1, 2, 3 };
        int* const p = a;

        cout << *(p++);
}
```

```
int main()
{
        int A[2][3] = { {1, 2, 3}, {4, 5, 6} };
        int* p1, * p2;

        int B[3] = { 7, 9, 0 };
        p1 = &B;

        p2 = A;

        cout << *p2;
}
```

```
int main()
{
```

`` void* vp; int a = 6; float b = 6.9; vp = &a; cout << *vp; vp = &b; cout << *vp; } ```	
`` int main() { void* vp; int a = 69; vp = &a; cout << (char*)vp << endl; cout << (int*)vp << endl; cout << (float*)vp << endl; } ```	
`` int main() { void* vp; float b = 6.9; vp = &b; cout << &b << endl; cout << &vp << endl; cout << (float*)vp << endl; cout << (float**)vp << endl; cout << (float***)vp << endl; cout << (float******)vp << endl; } ```	
`` int main() { void* vp; int a = 69; vp = &a; cout << &vp << endl; cout << &a << endl; cout << (void*)vp << endl; cout << *(void*)vp << endl; } ```	
`` int main() { void* vp; ```	

```
        int a = 69;

        vp = &a;
        cout << &a << endl;
        cout << &vp << endl;
        cout << *(int*)vp << endl;
        cout << *(int*)*&vp << endl;
        cout << (int*)&vp << endl;

}
```

```
int main()
{
        void* vp;

        int a = 69;

        vp = &a;
        cout << &a << endl;
        cout << &vp << endl;
        cout << *(char*)vp << endl;
        cout << (char*)vp << endl;
        cout << (char*&)vp << endl;
        cout << (char*&&)vp << endl;
        cout << *&(char*&)vp << endl;
        cout << (char**)vp << endl;
        cout << (void *)(char*)vp << endl;
        cout << (void *)(char***)vp << endl;
        cout << (char***)&vp << endl;
}
```

```
int main()
{
        void* vp;
        void** vvp = &vp;
        int a = 69;

        vp = &a;
        cout << &a << endl;
        cout << &vp << endl;
        cout << &vvp << endl;
        cout << *vvp << endl;

        cout << (char *)*vvp << endl;
        cout << (void*)(vvp) << endl;
        cout << (**vvp) << endl;
        cout << (char**)(*vvp) << endl;
        cout << &(*vvp) << endl;

        cout << *((char*)*vvp) << endl;
        cout << (void*)(*vvp) << endl;
        cout << (void*)(char*)(vvp) << endl;
        cout << (void*)(void*)(char**)(vvp) << endl;
        cout << (char**)(vvp) << endl;

}
```

```
int main()
{
        void* vp;
        void** vvp;
```

```
        int a = 69;
        int* ip = &a;
        vvp = &ip;


        vp = &a;
        cout << &a << endl;
        cout << &vp << endl;
        cout << &vvp << endl;
        cout << *vvp << endl;
        cout << *ip << endl;

}
```

```
int main()
{
        void* vp;
        int a = 69;
        int* ip = &a;
        void* & vvp = vp;


        vp = &a;
        cout << &a << endl;
        cout << &vp << endl;
        cout << &vvp << endl;
        cout << *(char *)vvp << endl;
        cout << (void *)ip << endl;
        cout << (void*)&ip << endl;

        cout << *(int *)vvp + (int **)vp << endl;
        cout << *(int*)vvp + *(int*)vp << endl;

}
```

```
int a = 5;
int b = 6;
int* p = &a;


int* ABC() {
        return &b;
}

int* DEF(int* p) {
        return p;
}

int& DEF() {
        return *p;
}

int& GHI() {
        return a;
}

int main()
{
        int a = 4;
        int* p;
```

```
	cout << *(ABC()) << endl;

	p = DEF(&::a);
	cout << *p << endl;

	DEF() = 1;
	cout << ::a << endl;

	a = GHI();
	cout << a << endl;
}
```

```
int main() {

	char pf[] = "PF is an interesting course";
	char* ptr = pf;

	cout << "1 " << ptr[3] << *ptr << endl;
	cout << "2 " << ++ptr << endl;
	cout << "3 " << ++(*pf) << endl;
	cout << "4 " << pf + 5 << endl;
	cout << "5 " << (ptr + 5)[-3] << endl;

}
```

```
int& mystery(int*& p)
{
	static int s = 3;

	if (p)
	{
		cout << *p + 2<< endl;
		delete[] p;
		p = nullptr;
	}

	p = new int[s] {s + s, s + 3, s + 2};

	return s;
}
int& magic(int* p)
{
	if (p)
	{
		cout << *p + 2 << endl;
		delete[] p;
		p = nullptr;
	}

	static int s = 3;
	p = new int[s] {s + s};

	return s;
}
int main()
{
	int* ptr = nullptr;

	mystery(ptr) = 5;
```

```
        cout << *ptr << endl;

        mystery(ptr)++;
        cout << *ptr << endl;

        magic(ptr) = 2;
        cout << *ptr << endl;

        if (!ptr)
                cout << "Ok that's All" << endl;
}
```

```
void mystery(int** p)
{
        *p += 2;
        cout << (*p)[1] << endl;
        cout << p[0][-3] << endl;

        p = new int* [3] {*p, *p - 2, *p - 3};

        cout << *(*p + 2) << endl;
        (*p + 3)[-1] = 20;

        delete[] p;

}

void magic(int ptr[][3])
{
        ptr += 1;
        cout << **ptr << endl;
        **ptr += 3;

        cout << ptr[-1][1] << endl;
}


int main()
{
        int arr[3][3] = { 1, 2, 3, 4,
                5, 6, 7, 8, 9 };

        magic(arr + 1);

        int* p = &arr[-1][6];
        mystery(&p);

        p += 2;
        for (int i = 0; i < 3; i++)
                cout << *(p - i) << endl;

}
```

```
int main()
{
        int s = 3, b = 4, t = 7;
        int* ptr = &s;
        int*& rptr = ptr;
```

```
        cout << *ptr << endl;
        cout << *rptr << endl;

        ptr = &b;
        cout << *ptr << endl;
        cout << *rptr << endl;

        int* nptr = &t;
        rptr = nptr;

        cout << *nptr << endl;
        cout << *rptr << endl;
        cout << *ptr << endl;

}
```

```
int main()
{
        int s = 3, b = 4, t = 7, & q = t;
        int* ptr = &s;
        int** rptr = &ptr;

        cout << *ptr << endl;
        cout << **rptr << endl;

        ptr = &b;
        cout << *ptr << endl;
        cout << **rptr << endl;

        q = b + 2;
        *rptr = &t;
        cout << **rptr << endl;
        cout << *ptr << endl;

        int* nptr = &s;
        rptr = &nptr;

        cout << *nptr << endl;
        cout << **rptr << endl;
        cout << *ptr << endl;

}
```

```
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);
}
```

```
int s = 3;

int func(int a) {

        if (a == 0)
                return func(s--);
        if (a < 0)
        {
                s -= 1;
                cout << s << " , " << a << endl;
                return s + a;
        }

        return func(func(a - 1) - 1);
}

int main() {
        func(2);
}
```

```
void doMagic(int arr[], int n)
{
   if (n <= 1)
      return;

   doMagic(arr, n - 1);

   int last = arr[n - 1];
   int j = n - 2;

   while (j >= 0 && arr[j] > last)
   {
      arr[j + 1] = arr[j];
      j--;
   }
   arr[j + 1] = last;
}

int main()
{
   int arr[] = { 12, 11, 13, 5, 6, 18};
   int n = sizeof(arr) / sizeof(arr[0]);

   doMagic(arr, n);
   for (int i = 0; i < n; i++)
      cout << arr[i] << " ";
}
```

```
int fun(int a, int b)
{
   if (b == 0)
      return 0;
   if (b % 2 == 0)
      return fun(a + a, b / 2);
```

`` return fun(a + a, b / 2) + a; } int main() { cout << fun(4, 3); return 0; } ```	
`` int fun(int n) { if (n > 100) return n - 10; return fun(fun(n + 11)); } int main() { cout << " " << fun(99) << " "; return 0; } ```	
`` void abc(const char *s) { if (s[0] == '\0') return; abc(s + 1); abc(s + 1); cout << s[0]; } int main() { abc("aneeq"); return 0; } ```	
`` int fun(int count) { cout << count << endl; if (count < 3) { fun(fun(fun(++count))); } return count; } int main() { fun(1); return 0; } ```	
`` int fun(int x, int y) { if (y == 0) return 0; return (x + fun(x, y - 1)); ```	

```
}

int main()
{
   cout << fun(2, 3);
   return 0;
}
```

```
int foo(int n, int r) {

   if (n > 0)
      return (n % r + foo(n / r, r));
   else
      return 0;
}

int main()
{
   cout << foo(345, 10);
   return 0;
}
```

```
void crazy(int n, int a, int b)
{
   if (n <= 0)
      return;
   crazy(n - 1, a, b + n);
   cout << n << " " << a << " " << b << endl;
   crazy(n - 1, b, a + n);
}

int main()
{
   crazy(3, 4, 5);
   return 0;
}
```

```
int okay(int n, int m, int PD[4][4])
{

   if (n == 1 || m == 1)
      return PD[n][m] = 1;

   if (PD[n][m] == 0)
   {
      PD[n][m] = okay(n - 1, m, PD)
         + okay(n, m - 1, PD);
   }

   return PD[n][m];
}

int main()
{
   int PD[4][4] = { 0 };

   cout << okay(3, 3, PD);
```

```
    return 0;
}
```

```
void swapIt(char& a, char& b)
{
    char c = a;
    a = b;
    b = c;
}

void foo(char* a, int l, int size)
{
    if (l == size)
        cout << a << endl;
    else {
        for (int i = l; i <= size; i++) {
            swapIt(a[l], a[i]);
            foo(a, l + 1, size);
            swapIt(a[l], a[i]);
        }
    }
}

int main()
{
    char str[] = "ABC";
    foo(str, 0, 2);
}
```

```
int fun(int a, int b)
{
    return a > b ? a : b;
}

int foo(int A[], int n)
{
    if (n == 1)
        return A[0];

    return fun(A[n - 1], foo(A, n - 1));
}

int main()
{
    int A[] = { 1, 4, 45, 6, -50, 10, 2 };
    int n = sizeof(A) / sizeof(A[0]);

    cout << foo(A, n);
    return 0;
}
```

```
void fun(char& a, char& b)
{
    char c = a;
    a = b;
    b = c;
}
```

```
void foo(char * str, int size, int i = 0 )
{
   if (i == size / 2)
      return;

   fun(str[i], str[size - i - 1]);
   foo(str, size, i + 1);
}

int main()
{
   char str[] = "maxe ruoy";
   int size = sizeof(str) / sizeof(str[0]);

   foo(str, size - 1);
   cout << str;

}
```

```
bool foo(int n, int i = 2)
{
   if (n <= 2)
      return (n == 2) ? true : false;
   if (n % i == 0)
      return false;
   if (i * i > n)
      return true;

   return foo(n, i + 1);
}

int main()
{
   int n = 35;
   if (foo(n))
      cout << "Yes";
   else
      cout << "No";

   return 0;
}
```

```
int find(int n)
{
   if (n == 0)
      return 0;
   else
      return (n % 2 + 10 * find(n / 2));
}

int main()
{
   int n = 11;
   cout << find(n);
   return 0;
}
```

```
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;
}
```

```
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(n, num);
	cout << endl;
	pattern(n - 1, count, num + 1);
}

int main()
{
	int n = 5;
	pattern(n, 1, 1);
	return 0;
}
```

Question 1

a) What would be the output produced by executing the following C++ code? Identify errors, if any (Either write output or error , Both will not be acceptable).

```
#include <iostream>
 using namespace std;
 class Number {
 private:
 int n;
 public:
 Number() : n(0) {
 cout << n;
 }

 Number( int nn ): n(nn)
 {
 cout << n;
 }

Number(Number const& otherNum): n(otherNum.n+1)
 {
 cout << n;
 }

 void display() { cout << n; }
 void increase() { n += 1; }
 };
 int main(){
 Number a, b(1), c(b);
 b.increase();
 c.display();
 b.display();
 }
```

01222

b) What would be the output produced by executing the following C++ code? Identify errors, if any (Either write output or error, both will not be acceptable).

```
#include <iostream>
 using namespace std;
 class Memory {
 float capacity;
 public:
 Memory(int cap = 1) {
 capacity = cap;
 cout << " Added Memory of Capacity= "
<< capacity << " G " << endl;
 }
 ~Memory() {
 cout << " Removed Memory of Capacity= "
 << capacity << " G " << endl;
 }
 };
 class Core {
 float speed;
 public:
 Core(float speed_ = 3.3) {
 speed = speed_;
 cout << " Added 1 Core of Speed= "
 << speed << " GHz " << endl;
 }
~Core() {
 cout << " Removed 1 Core of Speed= "
 << speed << " GHz " << endl;
 }
 };
 class Processor {
 const int ncores;
 Core cores[4];
 public:
 Processor() :
 ncores(4) {
 cout << " Added a Processor of "
 << ncores << " Cores " << endl;
 }
~Processor() {
 cout << " Removed a Processor of = "
 << ncores << " cores " << endl;
 }
 };
 class Mobile {
 Memory m;
 Processor p;
 public:
 Mobile() {
 cout << " Building a Mobile " << endl;
```

```
}
~Mobile() {
cout << " Destroying a Mobile " << endl;
}
};
int main() {
Mobile m;  cout << " :) The End " << endl; }
```

```
Added Memory of Capacity= 1 G
 Added 1 Core of Speed= 3.3 GHz
 Added 1 Core of Speed= 3.3 GHz
 Added 1 Core of Speed= 3.3 GHz
 Added 1 Core of Speed= 3.3 GHz
 Added a Processor of 4 Cores
 Building a Mobile
 :) The End
 Destroying a Mobile
 Removed a Processor of = 4 cores
 Removed 1 Core of Speed= 3.3 GHz
 Removed 1 Core of Speed= 3.3 GHz
 Removed 1 Core of Speed= 3.3 GHz
 Removed 1 Core of Speed= 3.3 GHz
 Removed Memory of Capacity= 1 G
```

c) What would be the output produced by executing the following C++ code? Identify errors, if any (Either write output or error, both will not be acceptable).

```
#include <iostream>
#include <cassert>
using namespace std;
class Number {
public:
static int n;
Number() {cout << n++<<endl; }

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

Number(Number const& otherNum){  cout << n<<endl; }

 ~Number(){cout<<--n;}
 };

void fun(Number n){
     cout<<n.n<<endl;
     n.somefunc();
 }
 int Number::n=0;

 int main(){
 Number a, b(9), c(a);
 fun(b);

 return 0;
 }
```

0
9
9
9
9
4321

d) What would be the output produced by executing the following C++ code? Identify errors, if any (Either write output or error, both will not be acceptable).

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

 private:  int *n;

 public:
    Integer() : n(new int) { *n=5;}

    Integer( int nn ):n(new int){   *n=nn;     cout << *n<<" ";   }

    Integer(Integer const& otherNum): n(otherNum.n){  cout << *n<<" ";   *n+=4;       }

     void display() { cout << *n<<" "; }

     void increase() { *n += 1; }  };

 int main(){

                Integer a, b(1), c(b);

                b.increase();

                c.display();

                b.display();  }
```

1 1 6 6

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

class A{
          int *p;
   static int a;

   public:
   A();
   A(int);
   A(A&);
   void display();
   void setter(int);

         };

int A::a = 7;

A::A()
{
 p= new int (a++);
 (*p)++;
 }

 A::A(int a)
{
 p= new int;
 *p = a;
}

A::A(A &temp)
{
  p= new int;
 *p = *(temp.p);
}

void A::display()
{
  cout << *(this->p) <<endl;

 }

 void A::setter(int k)
{
  *(this->p) = k;

 }

int main() {
  A obj1;
  A obj2 = obj1;
  obj1.display();
  obj2.display();

  obj2.setter(12);
  obj1.display();
  obj2.display();
  return 0;
}
```

8
8
8
12

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

class A{
          int *p;
   static int a;

   public:
   A();
   A(int);
   A(A&);
   void display();
   void setter(int);

         };

int A::a = 7;

A::A()
{
 p= new int (a++);
 (*p)++;
 }

 A::A(int a)
{
 p= new int;
 *p = a;
}
```

```
A::A(A &temp)
{
  p= new int;
 *p = *(temp.p);
}

void A::display()
{
  cout << *(this->p) <<endl;

 }

 void A::setter(int k)
{
  *(this->p) = k;

 }
int main() {
    A obj1;
  A obj2 = obj1;
  obj2.setter(12);

  A* obj3 = new A(obj2);
  obj3.setter(15);
  obj2 = obj3;
  obj1.display();
  obj2.display();
  obj3.display();
  return 0;
}
```

Error1 : obj3 is a pointer, cannot call setter function with . operator. Either call it with arrow notation obj3->setter(15) or with deference notation (*obj3).setter(15).

Error2: Cannot assign pointer to object. Obj2 is an object and Obj3 is a pointer.

Error3: obj3 is a pointer, cannot call display function with . operator. Either call it with arrow notation obj3->display() or with deference notation (*obj3). display()

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

int global = 9;

class A{
          int a,b;
   static int x;
   static int* p;

   public:
   A();
   A(int);
   A(int, int);
        };

int A::x = 19;
int* A::p = &global;

A::A()
{
 cout << 1<<endl;
 x+5;
 }
```

```
A::A(int a)
{
  this->a = a;
  (this->a)++;

  b = this->a;
  x--;
  cout << a <<endl;
  p = &(this->a);

}

 A::A(int a, int b)
{
  this->a = a;
  this->b = 2;
  cout << a << endl;
  x++;
  cout << this->b << endl;
  cout << *p;

}
int main() {
   A obj1(3),obj4;
  A obj2 = obj1;

  A* obj3 = new A(3,5);
  return 0;
}
```

```
3
1
3
2
4
```

PRACTICE PROBLEMS – Linked List

Q1)

```
struct Node {
   char data;
   Node* next;
};
int Mutate(Node* head, char d){
          Node* ptr = head;
          bool check = false;
   Node* last_start = NULL, *last_end = NULL, *pre = NULL;
   while(ptr != NULL){
      if(ptr->data == d){
         if(last_start){
            Node* temp = last_start->next;
            last_start->next = last_end->next;
            pre->next = temp;
            last_end->next = ptr;
            last_start = pre;
         }
         last_start = pre;
         last_end = ptr;
         while (ptr->data == d)
         {
            last_end = ptr;
            ptr = ptr->next;
         }

      }
      pre = ptr;
      ptr  =ptr->next;
   }
   Node* temp = last_start->next;
   last_start->next = last_end->next;
   pre->next = temp;
   last_end->next = ptr;
   last_start = pre;
   return 0;
}
```

Train = A -> C -> B -> C -> C -> A -> D -> C -> C -> C -> B -> NULL

What will be the resultant List train after passing Mutate(List, 'C');

What is Time Complexity?

Q2)

```
void func(Node* head){
   bool ex;
   Node* iNode = head;
   Node* cNode = NULL, *pNode = NULL;
   do{
      ex = false;
      while(cNode = iNode && cNode->next != pNode){
         iNode = iNode->next;
         if(cNode->data < iNode.data){
            int temp = cNode->data;
            cNode->data = iNode->data;
            iNode->data = temp;
            ex = true;
         }
      }
      pNode = iNode;
      iNode = head;
   }while(ex);

}
```

head = 9 -> 8 -> 7 -> 1 -> 5 -> NULL

What will be the list after passing func(head)?

Q3)

Write a print function that prints a node, then skips two nodes to print another node. For example, for the list

Head = 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 ->8

The function prints

1 -> 4 -> 7

Q4)

What will be the linked list after the following function?

Head = 6 -> 13 -> 11 -> 6 -> 13 -> 11 -> 5 -> 4 -> 5 -> 10

```
struct Node{
   int data;
   Node* next;
}
void func(Node* head){
   Node* ptr1, *ptr2, *nod;
   ptr1 = start;
   while(ptr1 && ptr1->next){
      ptr2 = ptr1;
      while (ptr2->next)
      {
         if(ptr1->data == ptr2->next->data){
            nod = ptr2->next;
            ptr2->next = ptr2->next->next;
            delete nod;
         }
         else{
            ptr2 = ptr2->next;
         }
      }
      ptr1 = ptr1->next;
   }
}
```

Q5)

Write a function in C++ which counts the duplicates in a linked List. For example, the list

7 -> 9 -> 7 -> 8 -> 9 -> NULL **will return 2.**

Q6)

```
int func(Node* head) {

   Node* ptr1 = head;
   Node* ptr = head;

   while (ptr != NULL && ptr->next != NULL) {
      ptr = ptr->next->next;

      ptr1 = ptr1->next;
   }
   return ptr1->data;
}
```

What will be the output if List = 1 -> 4 -> 5 -> 2 -> 9 -> 5 -> NULL is passed in function?

Practice Problems

1) **When will the Quicksort perform poorly (worst case)?**

2) **We have a system running insertion sort and we find that it's completing faster than expected. What could we conclude about the input to the sorting algorithm?**

3) **Explain which sorting algorithm you would use to sort the input array the fastest and why you chose this sorting algorithm. An array of n Comparable objects that is sorted except for k randomly located elements that are out of place (that is, the list without these k elements would be completely sorted)**

4) **Which sorting algorithm will have the best time complexity for sorting this array? Select one of them: Bubble | Selection | Insertion**

   a. **1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | -3 | -4**

   b. **1 | 15 | 3 | 4 | 5 | 8 | 10 | 12 | 13 | 2**

   c. **32 | 27 | 25 | 5 | 9 | 10 | 15 | 18 | 22 | 24**

5)

Given an array, apply **Shell Sort** on it:

9	8	3	7	5	6	4	1

```
void shellSort(int array[], int n)
{
        for (int interval = n / 2; interval > 0; interval /= 2)
      {
              for (int i = interval; i < n; i += 1)
            {
                    int temp = array[i];
                    int j;
                    for (j = i; j >= interval && array[j - interval] > temp; j -=interval)
                  {
                          array[j] = array[j - interval];
                    }
                    array[j] = temp;
              }
      }
}
```

1. Perform Dry run of the code and show array at each step. [10 marks]
2. Indicate the best and worst case of given algorithm in terms of Big -Oh [5 marks]

Solutions

1) **Quicksort has a worst case runtime of Θ(N2), if the array is partitioned very unevenly at each iteration.**

2) **Input already sorted.**

3) **Sort: Insertion sort**
   Runtime: O(nk)
   Explanation: For the n – k sorted elements, insertion sort only needs 1 comparison to check that it is in the correct location (larger than the last element in the sorted section). The re-maining k out-of-place elements could be located anywhere in the sorted section. In the worst case, they would be inserted at the beginning of the sorted section, which means there are O(n) comparisons in the worst-case for these k elements. This leads to an overall runtime of O(nk + n), which simplifies to O(nk).

4)
   a. **Insertion sort**
   b. **–**
   c. **--**

Q-02 /DLD/SP'23 – Sec B | 08/03/2023 | Marks: 10 / Time: 10 min

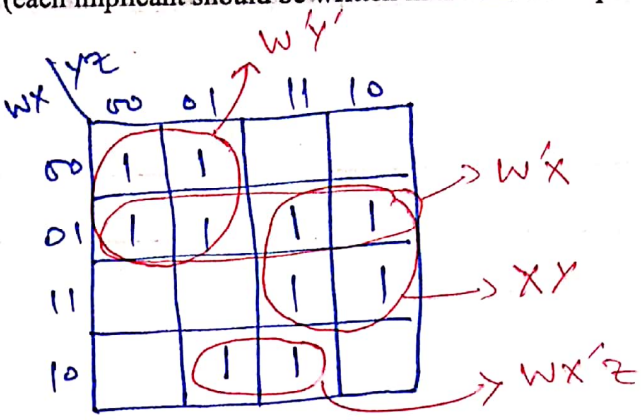
Name: _____ Roll No: _____ Section: _____

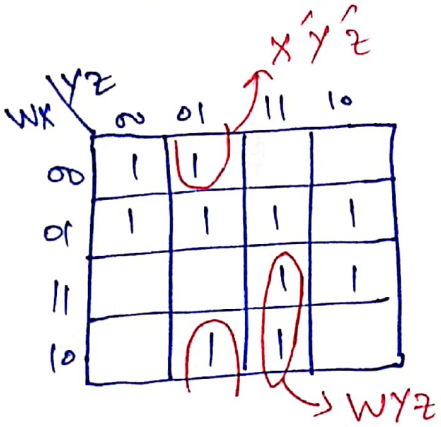
Question: Consider the following Boolean function.

$$F(W,X,Y,Z) = \sum(0,1,4,5,6,7,9,11,14,15)$$

a. List all the prime implicants (each implicant should be written in the form of a product term). **[4]**

wx \ yz	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10





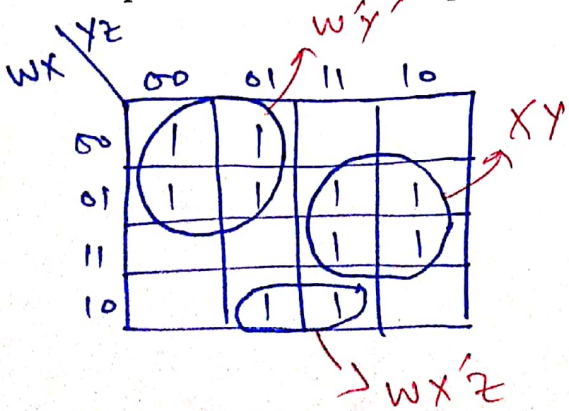
Prime Implicants are:-

w'y', w'x, xy, wx'z, x'y'z, wyz

b: List all of the essential prime implicants (again each implicant should be in the form of a product term) **[3]**

w'y', xy,

c. Write all possible minimal sum-of-products equations for this function. **[3]**



F = w'y' + xy + wx'z

Q-02 /DLD/SP'23 – Sec B | 08/03/2023 | Marks: 10 / Time: 10 min

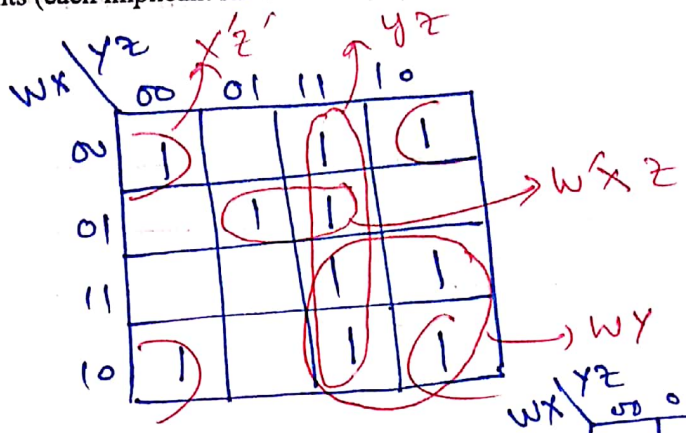
Name: _____ Roll No: _____ Section: _____

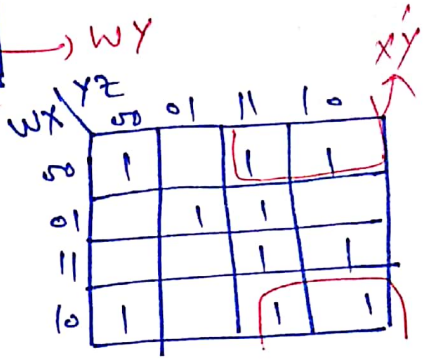
Question: Consider the following Boolean function.

$$F(W,X,Y,Z) = \sum(0,2,3,5,7,8,10,11,14,15)$$

a. List all the prime implicants (each implicant should be written in the form of a product term). **[4]**

wx \ yz	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10





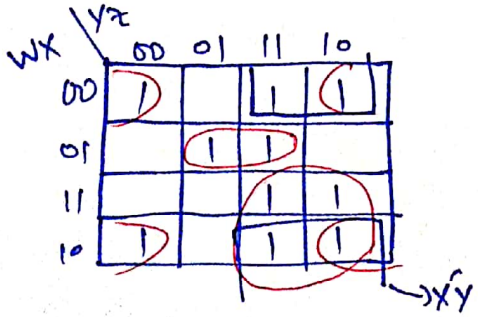
Prime Implicants are :-

x'z', yz, w'xz, wy, x'y

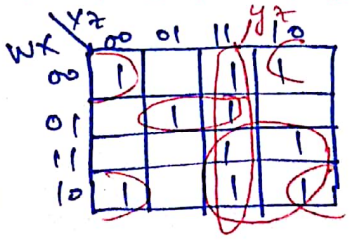
b. List all of the essential prime implicants (again each implicant should be in the form of a product term) **[3]**

x'z', w'xz, wy

c. Write all possible minimal sum-of-products equations for this function. **[3]**



F = x'z' + w'xz + wy + x'y → 1st Solution.



F = x'z' + w'xz + wy + yz → 2nd solution.

Q-02 /DLD/SP'23 – Sec C | 08/03/2023 | Marks: 10 / Time: 10 min

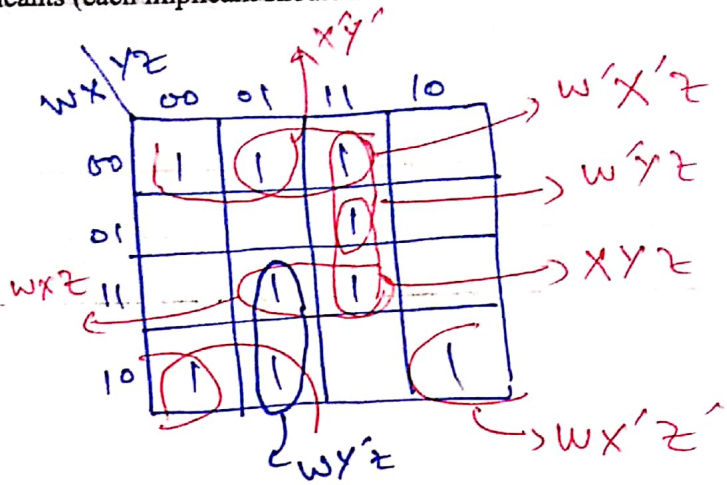
Name: _____ Roll No: _____ Section: _____

Question: Consider the following Boolean function.

$$F(W,X,Y,Z) = \sum(0,1,3,7,8,9,10,13,15)$$

a. List all the prime implicants (each implicant should be written in the form of a product term). **[4]**

wx \ yz	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10



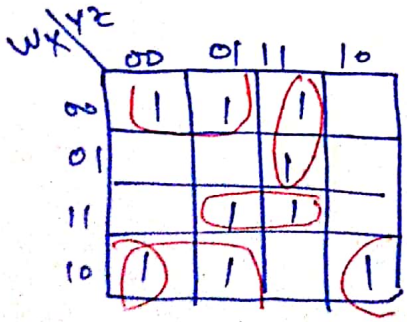
Prime Implicants are:-

x'y', w'x'z, w'yz, xyz, wxz, wx'z', wy'z

b. List all of the essential prime implicants (again each implicant should be in the form of a product term) **[3]**

x'y', wx'z'

c. Write all possible minimal sum-of-products equations for this function. **[3]**



F = x'y' + wx'z' + w'yz + wxz

Q-02/DLD/SP'23 – Sec C 08/03/2023 Marks: 10 / Time: 10 min

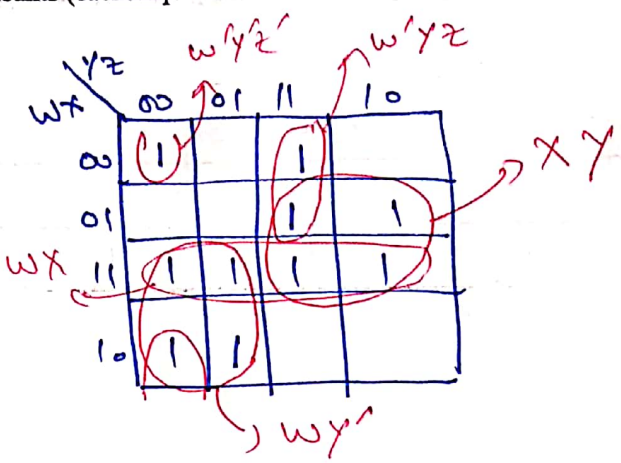
Name: _____ Roll No: _____ Section: _____

Question: Consider the following Boolean function.

$$F(W,X,Y,Z) = \sum(1,3,6,7,8,9,12,13,14,15)$$

a. List all the prime implicants (each implicant should be written in the form of a product term). **[4]**

wx \ yz	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10



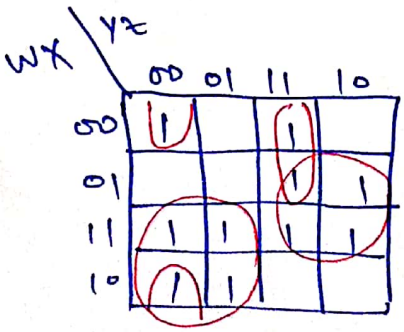
Prime Implicants are:-

w'y'z', w'yz, xy, wx, wy'

b. List all of the essential prime implicants (again each implicant should be in the form of a product term) **[3]**

w'y'z', w'yz, xy, wy'

c. Write all possible minimal sum-of-products equations for this function. **[3]**



F = w'y'z' + w'yz + xy + wy'

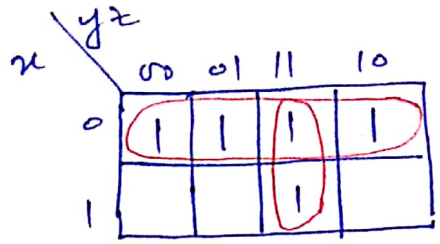
Q-02 /DLD/SP'23 – Sec D | 08/03/2023 | Marks: 10 / Time: 10 min

Name: Solution | Roll No: _____ | Section: _____

Question: Consider the following Boolean function.

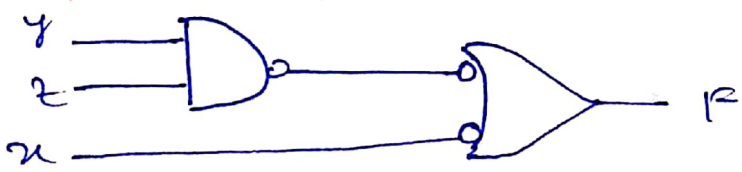
$$x'y' + yz + x'yz'$$

a. Simplify the Boolean function using three variable k-map: **[5]**



$F = x' + yz$

b. Implement the simplified function obtained in part **'a'** using two level NAND gate circuit. **[5]**



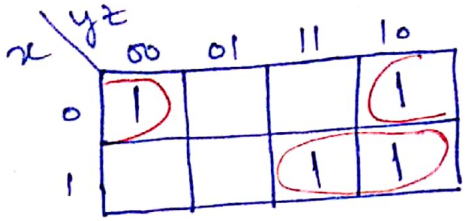
Q-02 /DLD/SP'23 – Sec D | 08/03/2023 | Marks: 10 / Time: 10 min

Name: _____ Roll No: _____ Section: _____

Question: Consider the following Boolean function.

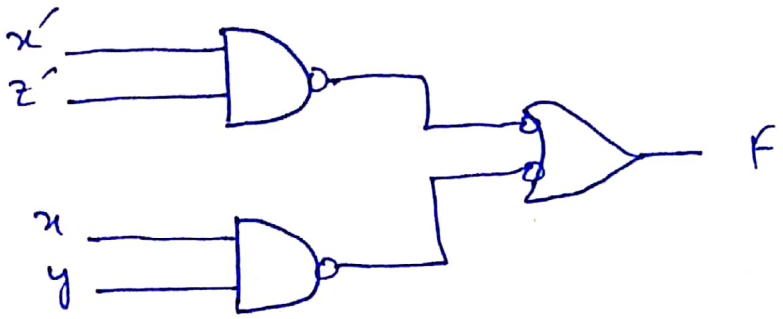
$$xy + x'y'z' + x'yz'$$

a. Simplify the Boolean function using three variable k-map: **[5]**



$F = x'z' + xy$

b. Implement the simplified function obtained in part **'a'** using two level NAND gate circuit. **[5]**



Name: **Quiz-3** **Roll No:**

Total Marks: 20 **Section: OOP-C** **Time: 15 mins**

Question-1[4-Marks]

Write the output of the following codes, **if any error(s) mention them and specify the reason. Reason of Error must be written to claim Marks.** *Both errors & output are not accepted*.

```
class Complex {
        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";
}
```

(No marks without Justification)

Output

Justification:

Question-2[10-Marks]

```
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;
        }
```

Output

```
        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++;
        C = ++A;
        C.print();
        (++(A++)).print();
        (A + C).print();
        B++.print();
}
```

Question-3[3-Marks]

```
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 << z + x  << endl << z + y <<
                        endl << z + 3 << endl;
        }
};
int main() {
        Dummy d(10);
        d.print();
}
```

Output

Question-4[3-Marks]

```
class Magic {
public:
        int* ptr;
        Magic(int n): ptr(new int(n)) {}
        friend void operator<<(ostream& out, Magic& m){
                for (int i = 0; i < 3; i++)
                        out << m.ptr[i] << endl;
        }
        void doIt(int s){
                for (int i = 0; i < 3; i++)
                        ptr[i] = s + i;         }
};
int main() {
        Magic A(3), B(A);
        B.doIt(3);
        A.doIt(5);
        cout << A << B;
}
```

Output

Name: **Quiz-4** **Roll No:**

Total Marks: 25 **Section: OOP-C** **Time: 15 mins**

Question-1[9-Marks]

Write the output of the following codes, if any error(s) mention them and don't write output. *Both errors & output are not accepted.*

```
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);
}
```

Output

Question-2[10-Marks]

```
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:
```

Output

```
        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-3[6-Marks]

```
class A {
        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 {
        int b;
        A a;
        A* aptr;
public:
        B() :b(0), aptr(nullptr) {
                cout << "B() called." << endl;
        }
        B(int x, A* objPtr) :a(x + 5), aptr(objPtr), b(x) {
                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();
}
```

Output

Name: **Quiz-5** **Roll No:**

Total Marks: 25 **Section: OOP-C** **Time: 15 mins**

Question-1[10-Marks]

Write the output of the following codes. **There are *NO* errors, in any of the Questions.**

`` class A { int a; public: A(int s = 0) : a(s) { cout << "HA " << a << endl; } void print(){ cout << a << endl; } ~A(){ cout << "Bye A" << endl; } }; class B : public A { int b; public: B(int n = 0) : b(n), A(9) { cout << "HB " << b << endl; } ~B() { cout << "bye B" << endl; } }; class C :public A { int c; public: C(int n = 0): c(n) { cout << "HC " << c << endl; } ~C(){ cout << "bye C" << endl; } }; class D : public B, public C { int d; public: D(int n = 0) :C(3), d(n), B(2) { cout << "HD " << d << endl; } ~D(){ cout << "bye D" << endl; } }; int main(){ D obj; } ```	***Output***

Question-2[8-Marks]

`` class Vehicle { public: Vehicle() { cout << "Vehicle() called.\n"; ~Vehicle() { cout << "~Vehicle() called.\n"; } }; ```	***Output***

```
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"; }
};
int main() {
        Vehicle* vehicles[3];
        vehicles[0] = new MotorCycle;
        vehicles[1] = new Car;
        vehicles[2] = new Vehicle;
        for (int i = 0; i < 3; i++)
                delete vehicles[i];
}
```

Question-3[7-Marks]

```
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* x2 = new C;
        if (x2)
                delete x2;
}
```

Output

(Bonus)[3-Marks]

In Question 1, if I execute the following, { obj.print(); } it generates an Error, why is it so ?

Answer:

Name: **Quiz-6** **Roll No:**

Total Marks: 25 **Section: OOP-C** **Time: 15 mins**

Question-1[15-Marks]

Consider the code given below and **Write the output. There are *NO* errors in this code.**
It is better to make the class Diagram first, for better understanding of class relationships.

```
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.Pro2();
        cr.Pro3();
        cp = &o;
        cp->Pro1();
        app = &d;
        app->Pro1();
        app->Pro2();
        cr1.Pro1();
        cr1.Pro2();
        cr1.Pro3();
}
```

Output:

Question-2[10-Marks]

Consider the code given below, Mark '✓' where the code will produce error. ***No marks*** **will be given in case of** ***Overwriting.***

```
class A {

int i;
public:
        A() {}
        virtual int output() = 0;
};
class B : public A {
        int j;
};
```

```
class C {
        int x;
public:
        int f(int a) { return x * a; }
protected:
        void setX(int a) { x = a; }
        int getX() { return x; }
};
class D : public C {
        int z;
};
```

```
1.  A objA;
2.  B objB;
3.  C objC;
4.  D objD;
5.  objC.f(5);
6.  B.output();
7.  cout << C.getX();
8.  D.setX(1);
9.  D.f(3);
10. objD.f(8);
```

Line No	Error	Line No	Error
1		6	
2		7	
3		8	
4		9	
5		10	

Write the output of the following

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

void recursiveFunction(int **&array, int &rows, int r, int c, int next =2 ,int iteration = 0) {

  if (iteration >= (rows * rows) - 1){
     return;
  }
  else {
     int row = r, col = c;
     r--;
     c++;

     if (r < 0)
        r = rows - 1;
     if (c > rows - 1)
        c = 0;

     if (array[r][c] != 0) {
        r = row + 1;
        c = col;
     }

     array[r][c] = next;
     next++;

     recursiveFunction(array,rows,r,c,next,iteration+1);

  }
}
```

```
void displayMatrix(int **&matrix,int rows,int columns){
  for(int i=0;i<rows;i++){
     for(int j =0; j< columns;j++){
        cout<<matrix[i][j]<<" ";
     }
     cout<<endl;
  }
}

int main() {
  int rows = 3;

  int **array = new int *[rows];
  for (int i = 0; i < rows; i++)
     array[i] = new int[rows];

  for (int i = 0; i < rows; i++)
     for (int j = 0; j < rows; j++)
        array[i][j] = 0;

  int column = rows / 2, row = 0;
  array[row][column] = 1;
  int next = 2;

  recursiveFunction(array,rows,row,column);
  displayMatrix(array,rows,rows);

  for(int i=0;i<rows;i++)
     delete array[i];
  delete[] array;

}
```

Write the output of the following

```
int main(){
  recursiveFunction(5,3);
}

void recursiveFunction(int height, int length, int row = 1,int i = 1, int j = 5, int iteration = 0, int k =1)
{

  if (row==1){
     j=height;
  }

  if (row>height) {
     return;
  }
  else {
     if (k==j || k==i) {
        cout << "*";
     }
     else {
        cout << " ";
     }
     if (k==height) {
        k = 0;
        iteration += 1;
     }
     if (iteration == length) {
        cout << endl;
        i++;
        j--;
        k = 0;
        row+=1;
```

```
            iteration = 0;
        }

        k++;

        recursiveFunction(height,length,row,i,j,iteration,k);
    }
}
```

Write a recursive function find that finds given target value in the array. If value is not found your function must return -1, otherwise it should return the index of array where the value was found. Your function prototype must be as follows:
int find(int array[], int length, int target);

Given a number n, write a recursive function whether it's prime number or not. Your function should return true or false.

Given a string, write a recursive function to count total number of consonants in it. Your function should return the count of consonants.

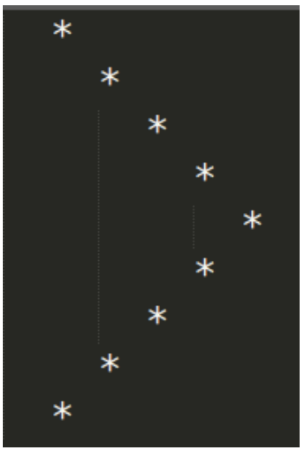
Given a string, write a recursive function to find the first uppercase letter of a String. Your function should return the letter.

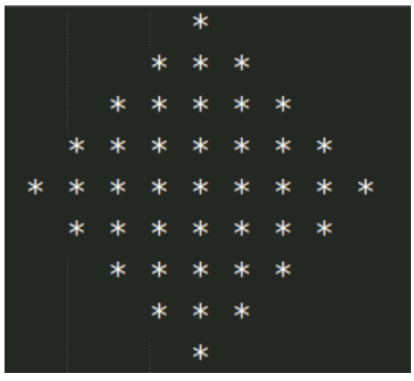
Given a decimal number as input, we need to write a recursive function to convert the given decimal number into equivalent binary number. Your function should return the binary equivalent of the number.

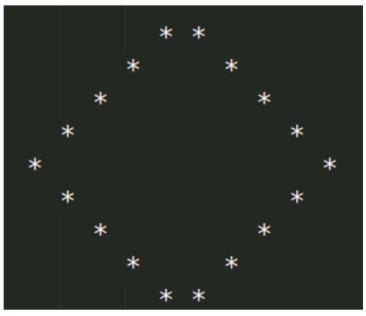
Write a function that accepts an array of integers and a number indicating the number of elements as arguments. The function should recursively calculate the sum of all the numbers in the array.

Write a function that uses recursion to raise a number to a power. The function should accept two arguments: the number to be raised and the exponent. Assume that the exponent is a nonnegative integer.

Write a C++ recursive function PrintPattern1 to print following pattern using recursion. No loops allowed whatsoever and you can write maximum two functions apart from main function. Your function prototype must be as follows: void PrintPattern1(int start, int end);







Write a recursive function that takes two matrix of same size and output whether they are equal or not. int equal(int ** matrix1, int** matrix2, int row, int column)

Write a program that takes a 2D pointer array and calculate sum of even and odd using recursive function. int sum(int **array, int row, int column, int &evenSum, int &oddSum)

Given following three values, the task is to find the total number of maximum chocolates you can eat.

1. money : Money you have to buy chocolates
2. price : Price of a chocolate
3. wrap : Number of wrappers to be returned for getting one extra chocolate.

It may be assumed that all given values are positive integers and greater than 1.

```
Input :   money = 16, price = 2, wrap = 2
Output :   15
Price of a chocolate is 2. You can buy 8 chocolates from
amount 16. You can return 8 wrappers back and get 4 more
chocolates. Then you can return 4 wrappers and get 2 more
chocolates. Finally you can return 2 wrappers to get 1
more chocolate.

Input :   money = 15, price = 1, wrap = 3
Output :   22
We buy and eat 15 chocolates
We return 15 wrappers and get 5 more chocolates.
We return 3 wrappers, get 1 chocolate and eat it
(keep 2 wrappers). Now we have 3 wrappers. Return
3 and get 1 more chocolate.
So total chocolates = 15 + 5 + 1 + 1

Input :  money = 20, price = 3, wrap = 5
Output :   7
```

Write functions for each of the following problems. Each problem should be solved by writing a recursive function (potentially with an auxiliary/driver function). Your final program **should not have** any loops in it. The main function of the file should demonstrate each of your solutions, by running some tests and producing some output. You are also required to write test cases

1. **Table of Squares:**
   Convert the following function to one that uses recursion:

```
void tableOfSquares (int n) {
        for (int num=1; num<=n; num++) {
                cout << num << " " << (num * num) << endl;
        }
}
```

2. **Recursive Power Function**
   Write a function that uses recursion to raise a number to a power. The function should take two argume
   for (int num=1; num<=n; num++) { nts, the number to be raised to the power (floating point) and the power (a non-negative int).

3. **isMember function**
   Write a boolean function named isMember that takes three arguments: an array, its size (number of elements) and a value. It should return true if the value is found in the array, or false if the value is not found in the array.

4. **Palindrome detector**
   A palindrome is any word, phrase, or sentence that reads the same forwards or backwards. Here are some palindromes (find more with google):

   - level
   - Pot top
   - A man a plan a canal Panama

Write a boolean function that determines if a string argument is a palindrome. The function should return true if the argument reads the same forwards and backwards. For full credit, your function should ignore spaces and be case-insensitive. Assume the input is just letters and spaces. Hint: you can do recursive calls on any substring of the original one (as long as it is shorter, and you have enough base cases).

Output:

Here is the sample. I'm sure you can do better:
Table of squares:

Enter a Number : 10

N	N Squared
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Power function:
2 to the 5th power: 32

Member function :
1 2 3 4 5 has 4?: YES
1 2 3 4 5 has 6?: NO

isPalindrome function :
Is "A man A plan A canal Panama" a palindrome? YES
Is "No one" a palindrome? NO

Activity no 02:

A) Write a program in C++ to print the Fibonacci series up to a given number using recursion.

B) Write a program in C++ to find the GCD (Greatest Common Divisor) of two numbers using recursion.

< Good Luck 😊 >

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();

}
```