BSCCS2005: Graded Assignment with Solutions
Week 1

1. Match the following:

[MCQ:2points]

A. Control Link	I. Region of the program where a variable is available for use
B. Activation Record	II. Pointer to the previous activation record
C. Scope	III. Duration/time during which a variable is available in the memory
D. Lifetime	IV. Stores the local variables

- ◯ A-II, B-I, C-IV, D-III
- √ A-II, B-IV, C-I, D-III
- ◯ A-II, B-IV, C-III, D-I
- ◯ A-I, B-IV, C-III, D-II

2. What will be the output of the following Python code?

[MCQ:2points]

```
class Demo:
    def __init__(self,str):
        self.name = str
    def print_Demo(self):
        print(self.name)

obj1 = Demo("IITM")
obj2 = Demo("Java")
obj1.print_Demo()
obj2.print_Demo()
```

- √ IITM
  Java
- ◯ IITM
  IITM
- ◯ Java
  Java
- ◯ Java
  IITM

3. What will be the output of the following Python code?

[MCQ:2points]

```
i = 42
def f():
    j = i+10
print(i)
f()
print(j)
```

- ◯ 42
  52
- ◯ 42
  0
- √ 42 followed by an error
- ◯ Error

4. Consider the Python code given below and choose the correct option.

[MCQ:2points]

```
def fun1(x):
    y = x+1
    def fun2(z):
        z = ((x+y)*(x-y))*z
        print(z)

    fun2(2)

fun1(5)
```

- ◯ x, y, z are stored in the same activation record.
- √ x and y both are stored in the same activation record, whereas z is stored in another activation record.
- ◯ x, y, z are each stored in different activation records.
- ◯ None of the above

Solution: In above code x, y are stored in activation record for function *fun1()* whereas z is stored in activation record of function *fun2()*

5. Consider the statements given below.
   Statement 1: **Player** is an object that has name, age and role, as its data.
   Statement 2: **Captain** is an object that has name, age, role, date of appointment as a captain, and number of years of experience, as its data.

   Identify the correct option regarding subtyping with respect to **Player** objects and **Captain** objects.

   [MSQ:2points]

   - √ Captain can be a subtype of Player.
   - ◯ Player can be a subtype of Captain.
   - ◯ Captain cannot be a subtype of Player.
   - √ Player cannot be a subtype of Captain.

Solution: Whatever the data stored by player object, same data also required to the captain.
Here, captain is also a player. Apart player, some extra data is required to the captain.
Captain object can reuse the data from player object by making captain object as subtype of player.
You cannot make player object as subtype of captain because all captain data may not be required to the player object.

6. Consider writing a 'low level' machine language program to perform the following operation:

$$d = a - b * c$$

Identify the correct order to execute the steps given below to perform the given operation.

1. Load the value from memory location a into register `R2`.
2. Load the values from memory locations b and c into registers `R1` and `R2` respectively.
3. Subtract the content of `R1` from `R2` and store the result back into `R1`.
4. Multiply the contents of registers `R1` and `R2` and store the result back into `R1`.
5. Store the content of `R1` into memory location d.

[MCQ: 2points]

- ◯ `2->1->4->3->5`
- √ `2->4->1->3->5`
- ◯ `1->2->4->3->5`
- ◯ `2->4->5->1->3`

Solution: The correct steps are as follows:

1. Load the values from memory locations `b` and `c` into registers `R1` and `R2` respectively.
2. Multiply the contents of registers `R1` and `R2` and store the result back into `R1`.
3. Load the values from memory location `a` into register `R2`.
4. Subtract the content of `R1` from `R2` and store the result back into `R1`.
5. Store the content of `R1` into memory location `d`.

7. Identify the most appropriate segregation of the following features between (A) static and (B) dynamic typing in the context of programming.

   1. A name in the program derives its data type from the assigned value.
   2. Every name needs to be declared with its type in advance.
   3. An uninitialized name has no type.
   4. A name cannot be assigned to an incompatible value (a value whose type is not compatible with the type of the name).
   5. A name can be assigned to any value, and the type of the value determines the type of the name.
   6. It does not allow any name to be assigned unless it is already declared explicitly with type.
   7. During any assignment, it cannot identify either if it is an assignment for a new name or if it is a reassignment for an existing name.

[MCQ: 2points]

- √ (A) - 2, 4, 6
  (B) - 1, 3, 5, 7
- ◯ (A) - 1, 3, 5
  (B) - 2, 4, 6, 7
- ◯ (A) - 1, 3, 4, 5
  (B) - 2, 4, 7
- ◯ (A) - 2, 4, 6, 7
  (B) - 1, 3, 5

Solution: The features for static type:

- Every name need to declared with their types in advance.
- A name cannot be assigned to an incompatible value (a value whose type is not compatible with the type of the name).
- It does not allow any name to be assigned unless it is already declared explicitly with type.

The features for dynamic type:

- A name in the program derives its data type from the assigned value.
- An uninitialized name has no type,

- A name can be assigned to any value, and the type of value determines the type of the name.
- During any assignment, it cannot identify either it an assignment for a new name or it is a reassignment for an existing name.

8. Consider a polyclinic which has a number of doctors. The doctors have schedules for their visiting days and times. Doctors can update their profiles and change their visiting times. Patients need to register in order to seek appointments with the doctors. Doctors provide prescriptions of various medicines to the patients.

   Given the above scenario, match the abstract types with the associated set of operations.

   [MCQ: 2points]

   ○
   - Doctor type with the operations – add and modify own profile, add and modify visiting timing, seek appointment
   - Prescription with the operation – write prescription
   - Medicine type with the operation – add medicines

   ○
   - Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription, seek appointment
   - Patient type
   - Prescription type with the operation – write prescription
   - Medicine type with the operation – add medicines

   √
   - Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription
   - Patient type with the operation – seek appointment
   - Prescription with the operation – add medicines
   - Medicine type

   ○
   - Doctor type with the operations – add and modify own profile, write prescription
   - Patient type with the operations – seek appointment, add and modify visiting timing,
   - Prescription with the operation – add medicines

Solution: The appropriate set abstract types associated with most proper related set of operations are:

- Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription
- Patient type with the operation – seek appointment
- Prescription with the operation – add medicines
- Medicine type

9. What will be the value of **num** after execution of the following code?

[MCQ: 2points]

```
def elementSum(n):
    sum = 0
    while (n != 0):
        sum = sum + n % 10
        n = n // 10
    return sum
num = 22
x = elementSum(num)
```

- ◯ 0
- √ 22
- ◯ 4
- ◯ 2
- ◯ 10

Solution: **num**, **n** are the names identifying to different memory location, so changing **n** has no effect on **num**

10. What will be the value of **Dict** after execution of the following code?

[MCQ: 2points]

```
def updateDict(d):
    d["rollno"] = 12
Dict = {"name" : "John", "Age" : 21}
updateDict(Dict)
```

- ◯ {"rollno" : 12, "Age" : 21}
- ◯ {"rollno" : 12}
- ◯ {"name" : "John", "Age" : 21}
- √ {"name" : "John", "Age" : 21, "rollno" : 12}

Solution: Dict, **d** are referring to the same memory location, so changing **d** also affects **Dict**

graphixs

BSCCS2005: Graded Assignment Questions with Solutions
Week {2}

{Write general instructions here}

1. What is the value of `str2` at the end of execution of the following Java code? [MCQ:2points]

```
public class StringExample {
    public static void main(String[] args) {
        String str1, str2;
        str1 = "welcome to IITM";
        str2 = str1.substring(0, 11) + "java course" ;
    }
}
```

- ◯ `java course`
- ◯ `welcome to IITM java course`
- √ `welcome to java course`
- ◯ `Error in code`

Solution: str1.substring(0, 11) gives "welcome to "
+ is used for string concatenation
"welcome to " + "java course" results in "welcome to java course"

2. What is the output of the following Java code ? [MCQ:2points]

```
public class Print {
    public static void main(String[] args) {
        int[] a = {1, 2, 3};
        int[] b = {1, 2, 3};
        System.out.println(a == b);
    }
}
```

- ◯ `a == b`
- √ `false`
- ◯ `true`
- ◯ `{1, 2, 3} == {1, 2, 3}`

Solution: It prints false. The "==" operator compares whether a and b are referring to the same memory location.

3. Consider the Java program below. [MCQ:2 points]

```
class FClass{
    public static void main(String[] args) {
        int i1 = 10, i2 = 29;
        double d;
        d = i2 / i1;
        System.out.print(d + " ");
        d = (double)(i2 / i1);
        System.out.print(d + " ");
        d = (double)i2 / i1;
        System.out.print(d);
    }
}
```

What will be the output?

- ◯ `2.9 2.9 2.9`
- ◯ `2 2.9 2.9`
- √ `2.0 2.0 2.9`
- ◯ `2.0 2.0 2.0`

Solution: In statement `d = i2 / i1;`, since `i2` and `i1` are `int`, it is an integer division and assigns `d` with 2. Since `d` is a `double`, it is printed as 2.0.
In statement `d = (double)(i2 / i1);`, since `i2` and `i1` are `int`, it is an integer division and the result get type-casted to `double`. Thus, `d` would be assigned to 2.0.
In statement `d = (double)i2 / i1;`, `i2` which is of `int` type, type-casted to `double`. Thus, the division becomes floating-point division and `d` would be assigned to 2.9.

4. Identify the correct definition(s) of a `boolean` variable named `flag` in Java from the following. [MSQ:2 points]

   ◯ `boolean flag = 1;`

   √ `boolean flag = true;`

   ◯ `boolean flag = TRUE;`

   ◯ `boolean flag = "false";`

Solution: In Java a `boolean` variable can be assigned to either `true` or `false`.

5. Consider the Java program below. [MCQ:2 points]

```
class Point{
    private int x;
    private int y;
    public Point(int x, int z) {
        x = x;
        y = z;
    }
    public void printPint() {
        System.out.println("(" + x + ", " + y + ")");
    }
}
class CClass{
    public static void main(String[] args) {
        Point p = new Point(10, 20);
        p.printPint();
   }
}
```

What will be the output/error?

- ◯ `(0, 0)`
- ◯ `(10, 0)`
- √ `(0, 20)`
- ◯ `(<garbage-value>, <garbage-value>)`
- ◯ `Compiler error: Invalid assignment in constructor`

Solution: In the constructor `public Point(int x, int z)`, for the statements:

```
x = x;
y = z;
```

Here the name of function parameter and data member is same for `x` so inside the constructor `x` refers to the local function parameter and not the object's data member. Such is not the case with the other parameter `z` and hence the initialization for `y` works in usual manner.

6. Consider the Java program below. [MCQ:2 points]

```
import java.util.*;
class Example{
    public static int doSomething(int num) {
        int n = num;
        int total = 0;
        for(int i = 1; i <= n; i++) {
            if (n % i == 0) {
                total = total + i;
            }
        }
        return total;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int number = sc.nextInt();
        int x = doSomething(number);
   }
}
```

What will `x` represent?

- ◯ Number of multiples of `number`
- ◯ Sum of multiples of `number`
- ◯ Number of factors of `number`
- √ **Sum of factors of** `number`
- ◯ Compiler Error

Solution:

- Program execution begins from `main()` function.
- In main function, user inputs `number`.
- `doSomething()` method is called by passing `number`
- In `doSomething()` value of `number` is assigned to `n`, `i` is initialized to 1 and is incremented in each iteration till `i = n`
- `i` is added to `total`, if `i` divides `n` without any reminder. That means `i` must be factor of `n`.
- Hence, `total` represents sum of factors of `n`.
- `doSomething()` returns `total` and is stored in `x`

7. Consider the Java code given below. [MCQ : 2 points]

```
class  WhileEx2
{
    public static void main(String[] args)
    {
        int i=0;
        while(i>0)
        {
            System.out.println(i);
        }
        do
        {
            System.out.println(i);
        }while(i>0);
    }
}
```

Choose the correct option regarding the given code.

- ◯ Generates no output but program compiles successfully.
- ◯ Generates a compilation error
- ◯ Generates output :
  0
  0
- √ **Generates output : 0**

Solution: In `while` loop, the body of the loop executes when condition is `true`, whereas in `do-while` loop, first time the loop-body always executes once and at the end of the loop condition would be checked.
In above code, `while` loop prints nothing since the condition is failed at the first time, but `do-while` prints 0 as the condition is checked at the end of the loop.

8. Consider the Java code given below. [MCQ : 2 points]

```
class Sample
{
    public static void main (String args[])
    {
        System.out.println(10+20+"IIT Madras");
        System.out.println("IIT Madras"+10+20);
    }
}
```

What will be the output/error?

- ◯ Compilation fails due to `System.out.println(10+20+"IIT Madras");`
- ◯ Compilation fails due to `System.out.println("IIT Madras"+10+20);`
- √ `30IIT Madras`
  `IIT Madras1020`
- ◯ Exception at run time.

Solution: `System.out.println(10+20+"IIT Madras");`
Here both `10` and `20` are integers `10+20` will generate `30`, now `30` integer and a string `IIT Madras` both are concatenated and it generates `30IIT Madras as output`.
`System.out.println("IIT Madras"+10+20)`
Here a string `IIT Madras` is concatenated with an integer `10` and it generates a string `IIT Madras10`, now a string `IIT Madras10` and an integer `20` are concatenated and it generates string `IIT Madras1020` as output.

9. Match the following. [MCQ:2 points]

```
A. System            I.   Prints arguments in a new line.
B  out               II.  Public class.
C. println()         III. Stream object.
D. braces {...}       IV.  Delimits blocks and statement.
```

- ◯ A-III, B-II, C-IV, D-I
- ◯ A-II, B-I, C-III, D-IV
- √ **A-II, B-III, C-I, D-IV**
- ◯ A-I, B-IV, C-II, D-III

Solution:

`System`: A Java program is a collection of classes. System is a public class in Java.
`out`: It is a stream object defined in the System class.
`println()`: Prints arguments in new line like python print()
braces {...} : Delimits blocks and statement. This is similar to the indentation in Python programming.

10. How is it possible to run the main method in Java without creating an object? [MCQ:2 Points]

- ◯ Java is an object oriented programming language. Thus we need to create an object of the main class, and call our `main()` method to run. Hence the question is fallacious.
- √ **The modifier static helps the `main()` method to run independently without creating an object.**
- ◯ The access modifier public helps the `main()` method to run independently without creating an object.
- ◯ The `main()` method is an exception. It is the only method that exist independently without the dynamic creation of an object.

Solution:

The `main()` method in java is declared static. Static methods are the methods in Java that can be called without creating an object of class.

11. Consider the Java code given below. [MCQ:2 points]

```
class FClass
{
    public static void main(String args[])
    {
        int arr[] = {0 , 1, 2, 3, 4, 5, 6, 7, 8, 9};
        int n = 9;
        n = arr[arr[n] / 2];
        System.out.println(arr[n] / 3);
    }
 }
```

Choose the correct option regarding the given code.

- ◯ The program generates output:
  `0`
- √ **The program generates output:**
  `1`
- ◯ The program generates output:
  `1.3333`
- ◯ The program generates a compilation error.

Solution:
We have `int n=9;`
`arr[arr[9]/2] = arr[9/2] = arr[4] = 4`.
Hence, `arr[4]/3 = 4/3 = 1`.

12. Consider the Java code given below. [MCQ:2 points]

(a)
```
public class FClass
{
    public static void main(String args[])
    {
        int x; x = 1;
        if(x) { System.out.println(x); }
        else { System.out.println("2"); }
     }
}
```

Choose the correct option regarding the given code.

- ◯ The program generates output:
  `0`
- ◯ The program generates output:
  `1`
- ◯ The program generates output:
  `2`
- √ **This program will generate an error.**

(b)
```
public class FClass
{
    public static void main(String args[])
    {
        int x=1;
        if(false) { System.out.println(x); }
        else { System.out.println("True"); }
    }
}
```

- √ **The program generates output:**
  `True`
- ◯ The program generates output:
  `False`
- ◯ The program generates output:
  `1`
- ◯ This program will generate an error.

Solution:
(a) `if` is a conditional statement. It accepts `boolean` as a parameter. Here `x` is an integer so this program will yield an error.
(b) If the condition is `false`, then the `else` block gets executed.

13. Consider the Java code given below..

[MCQ:2 points]

```
public class FClass
{
    public static void main(String args[])
    {
        switch(2)
        {
            case 1:
                System.out.println("One");
            case 2:
                System.out.println("Two");
            case 3:
                System.out.println("Three");
            default:
                System.out.println("Default");
                break;
        }
    }
}
```

What will be the output?

- ◯ `One`
- ◯ `Two`
  `Default`
- ◯ `Three`
- √ **None of them**

Solution: The output of the above program is:

```
Two
Three
Default
```

Because in between cases there is no break statement.

14. Match the following. [MCQ:2 points]

```
A. Multiple constructors    I.  constructor with empty arguments
B  Default constructor      II. Refers to the current class object
C. Copy constructors        III.Create a new object from an existing one
D. this                     IV. Overloading
```

- √ **A-IV, B-I, C-III, D-II**
- ◯ A-III, B-I, C-I, D-II
- ◯ A-IV, B-II, C-III, D-I
- ◯ A-IV, B-III, C-I, D-II

Solution:
Multiple constructors — Constructor overloading
Default constructor- If no constructor is defined, compiler creates its own. They do no have any arguments
Copy constructor — make a copy of an existing object
this — refers to the current class object

15. Considering the following Java code, choose the correct statement from among the given options regarding this code.

[MCQ:2points]

```
class Employee
{
    int eid;
    String ename;

    public void display()
    {
        System.out.println("eid: "+eid);
        System.out.println("ename: "+ename);
    }
}

class FClass
{
    public static void main(String[] args)
    {
        Employee e1 = new Employee();
        e1.display();
    }
}
```

- ◯ The code results in a compilation error as constructor is not defined.
- ◯ Program runs successfully, and the data members `eid` and `ename` contains random garbage value.
- √ **Program runs successfully, and the data members `eid` and `ename` contains *0* and *null* respectively.**
- ◯ None of the above options are correct.

Solution:
If no constructor is defined in a class the default constructor would be executed when an object is created. In JAVA uninitialized variables have values *0* for numeric types and *null* for string types.

BSCCS2005: Graded Assignment
Questions with Solutions
Week 3

1. Consider the following code: [Nalini: MCQ: 2points]

```
public class A {
    public void display( ){
        System.out.print("Hii ");
    }
}
public class B extends A{
    public void display(String s){
        display();
        System.out.println(s);
    }
    public static void main(String[] args) {
         A a = new B( );
        // Line 1
    }
}
```

What is the correct instruction to be written in *Line 1* in order to print `Hii Ram` as output?

- ◯ a.display("Ram");
- ◯ a.display("Hii Ram");
- ◯ ((A)a).display("Hii Ram")
- √ **((B)a).display("Ram");**

Solution:

- To print Hii Ram display method of subclass must be called.
- But `a` is a reference variable of type A
- So `a` must be made of type B, this can be done by type casting.
- **type casting is happening on reference variable not on the object**
- And then the display( ) method of subclass should be called

Consider the following code and answer the questions 2 and 3.

```
public class Polygon{
    public static void perimeter( ){
        System.out.print("In Polygon perimeter");
    }
    public void area( ){
        System.out.println("In Polygon area");
    }
    public void sides( ){
         System.out.println("In Polygon sides");
    }
    public void angleSum( ){
        System.out.println("In Polygon angleSum");
    }
}
public class Pentagon extends Polygon{
    public static void perimeter( ){
        System.out.println("In Pentagon perimeter");
    }
    public void area( ) {
         System.out.println("In Pentagon area");
    }
    public int sides( ){
         return 5;
    }
    public void angleSum(int x) {
        System.out.println("In Pentagon angleSum");
    }
}
```

2. Which method/s override/s methods in the parentclass?

[Nalini: MSQ: 2points]

- ◯ perimeter
- √ **area**
- ◯ sides
- ◯ angleSum

Solution:

- Method signature must be same in both parentclass and subclass to override the method

3. Which method/s is/are new methods of the subclass?

[Nalini: MSQ: 2points]

- ◯ perimeter
- ◯ area
- √ **sides**
- √ **angleSum**

4. Consider the following code: [Nalini: MSQ: 2points]

```
1    public class Employee {
2        public void display( ){
3            System.out.print("In Employee class");
4        }
5    }
6    public class TeamLead extends Employee{
7        public void display( ){
8            System.out.println("In TeamLead class");
9        }
10   }
11   public class Manager extends Employee, TeamLead{
12        public void display( ){
13            System.out.println("In Manager class");
14        }
15        public static void main(String[] arg) {
16            TeamLead t = new Employee( );
17        }
18   }
```

Identify the line/s which has/have error.

- ◯ Line 1
- ◯ Line 2
- ◯ Line 6
- ◯ Line 7
- √ **Line 11**
- ◯ Line 12
- ◯ Line 15
- √ **Line 16**

Solution:

- A class cannot inherit two classes, multiple inherits is not allowed in java.
- An object of type parent class cannot refer subclass. Because the parent class may not have all the functionalities of subclass

5. Consider the Java code given below. [ARUP: MCQ: 2 points]

```
public class Bird{
    public void fly(){
        System.out.println("it can fly");
    }
}
public class Duck extends Bird{
    public void swim(){
        System.out.println("it can swim");
    }
}
public class FClass{
    public static void doIt(Bird b){
        b.fly();
        if(b instanceof Duck)
            ((Duck) b).swim();
    }
    public static void main(String[] args){
        Duck d = new Duck();
        doIt(d);
    }
}
```

What will be the output?

- ◯ `it can fly`
- ◯ `it can swim`
- √ `it can fly`
  `it can swim`
- ◯ It generates compiler error

Solution:

In function call `doIt(d);`, `d` is of `Duck` type and implicitly type casted to the parent type `Bird`. However, `b` refers to a subclass object, i.e. object of `Duck`. Thus, `b.fly()` prints `it can fly`. The condition `b instanceof Duck` returns `true` and `((Duck) b).swim()` prints `it can swim`.

6. Consider the Java code given below. [ARUP: MCQ: 2 points]

```
public class Shape{
    public void area(){
        System.out.println("area is unknown");
    }
    public void volume(){
        System.out.println("volume is unknown");
    }
}
public class Rectangle extends Shape{
    public void area(){
        System.out.println("area of Rectangle");
    }
}
public class Cube extends Shape{
    public void area(){
        System.out.println("area of Cube");
    }
    public void volume(){
        System.out.println("volume of Cube");
    }
}
public class FClass{
    public static void compute(Shape s){
        s.area();
        s.volume();
    }
    public static void main(String[] args){
        Rectangle r = new Rectangle();
        Cube c = new Cube();
        compute(r);
        compute(c);
    }
}
```

What will be the output?

- ◯ `area is unknown`
  `volume is unknown`
  `area is unknown`
  `volume is unknown`
- ◯ `area of Rectangle`
  `area of Cube`
  `volume of Cube`

√ `area of Rectangle`
`volume is unknown`
`area of Cube`
`volume of Cube`

◯ It generates compiler error

Solution: For the call `compute(r);`, although the static type of `s` is `Shape`, the dynamic type is `Rectangle` (since it refers to object of `Rectangle`). Thus, it prints

```
area of Rectangle
volume is unknown
```

For the call `compute(c);`, although the static type of `s` is `Shape`, the dynamic type is `Cube` (since it refers to object of `Cube`). Thus, it prints

```
area of Cube
volume of Cube
```

7. Consider the following abstract types.

   - `Scanner`, with method `scanDocuments()`.
   - `Printer`, with method `printDocuments()`.
   - `Copier`, with methods `scanDocuments()` and `printDocuments()`.

   Identify the correct subtype and inheritance relationships between the classes.

   [ARUP: MSQ: 2 points]

   ◯ `Scanner` and `Printer` are subtypes of `Copier`

   √ `Copier` **is a subtype of** `Scanner` **and** `Printer`

   √ `Scanner` **and** `Printer` **both inherit from** `Copier`

   ◯ `Copier` inherits from both `Scanner` and `Printer`

   Solution: Since `Copier` has more functionality than `Scanner` and `Printer`, `Copier` is a subtype of `Scanner` and `Printer`.
   Since we can suppress one of the functions in `Copier` and use it as a `Scanner` or `Printer`, both `Scanner` and `Printer` inherit `Copier`.

8. Consider the Java code given below. [ARUP: MCQ: 2 points]

```
public class Employee{
    private int empid;
    private String name;

    public Employee(int empid_, String name_){
        empid = empid_;
        name = name_;
    }
    public Employee(){
        this(0, "unknown");
    }
    public void print(){
        System.out.print(empid + " : " + name + " : ");
    }
}
public class Manager extends Employee{
    private String department;

    public Manager(int empid_, String name_, String department_){
        _________________________   //line:1
        department = department_;
    }
    public void print(){
        super.print();
        System.out.println(department);
    }
}
public class FClass{
    public static void main(String[] args){
        Manager m = new Manager(101, "Nutan", "HR");
        m.print();
    }
}
```

Identify the appropriate option to fill in the blank at `line:1` such that output of the code will be:

`101 : Nutan : HR`

- ◯ `Employee(empid_, name_);`
- ◯ `this(empid_, name_);`
- √ `super(empid_, name_);`
- ◯ `empid = empid_; name = name_;`

Solution: Since the instance variables `empid` and `name` are declared as `private` in `class Employee`, they are invisible in `class Manager`. Thus, option-4 is incorrect. The only way to initialize the `private` instance variables in `class Employee` by calling it's constructor, which can be done by the subclass `Manager` using `super` keyword.

9. Consider the following code. [Bhaskar:MSQ:2points]

```
public class Student{
    public String sname;
    public String sid;
    public int sclass;

    public Student(String s_name,String s_id, int s_class){
        this.sname = s_name;
        this.sid =s_id;
        this.sclass = s_class;
    }
    public void display() {
        System.out.println("name:"+sname);
        System.out.println("id:"+sid);
        System.out.println("class:"+sclass);
    }
}

public class Toppers extends Student{
    public int marks;

    public Toppers(int marks){
        this.marks = marks;
    }

    public void display() {
        super.display();
        System.out.println("marks:"+marks);
    }
}

public class FClass{
    public static void main(String[] args){
        Toppers t = new Toppers(30);
        t.display();
    }
}
```

Choose all the correct option/s which mention/s the required modification (if any) in the code for successful compilation.

◯ `super.display` must be removed.

√ **Define an appropriate no argument constructor of Student class.**

√ **Call the Student class parameterized constructor explicitly inside Topper class's constructor.**

◯ No modification required, code compiles successfully.

Solution: While we define an object of Topper class the constructor of Topper class is executed. This constructor will first call the no argument constructor of its superclass implicitly if an explicit call to superclass constructor is missing.
We have defined a parameterized constructor in the superclass which hides the default no argument constructor and therefore the no argument constructor call by the child class constructor will not get resolved and subsequently generate an error.
There can be two solutions to this problem:

1. Defining a no argument constructor in superclass explicitly
2. Call the parameterised constructor (already defined) of superclass explicitly, inside the constructor's body of child class

10. Consider the following code. [Bhaskar:MCQ:2ppoints]

```
1.  public class Mammal{
2.     public String name;
3.     public int lifespan;
4.
5.     public Mammal(){
6.          name = "Tiger";
7.          lifespan = 45;
8.     }
9.
10.    public void show() {
11.         System.out.format("name = %s : lifespan = %d",name,lifespan);
12.    }
13.    public void display() {
14.         System.out.println("Mammal details");
15.    }
16. }
17.
18. public class Endangered extends Mammal{
19.     public boolean endanger_status;
20.
21.     public Endangered(){
22.         endanger_status = false;
23.     }
24.
25.     public void show() {
26.         System.out.println("endanger status of "+
                                this.name + " is " + endanger_status);
27.     }
28.     public void display(String species) {
29.         System.out.println("Endanger status");
30.     }
31. }
32.
33. public class FClass{
34.     public static void main(String args[]) {
35.         Mammal m1 = new Endangered();
36.         m1.show();
37.         m1.display();
38.     }
39. }
```

Choose the correct option regarding the given code.

- ◯ This code produces a compilation error on line 37
- ◯ This code produces a compilation error on line 36
- ◯ This code will generate the output:
  endanger status of Tiger is false
  Endanger status
- √ **This code will generate the output:**
  endanger status of Tiger is false
  Mammal details
- ◯ This code will generate the output:
  endanger status of *null* is false
  Mammal details

Solution: The `display()` and `display(String)` are two overloaded methods in `Endangered` class whereas the `show()` method of `Mammal` class is an overridden method.
`Overloading` is done during compile time and method invocation is resolved depending upon reference type whereas for overridden methods instance type decides the version to be invoked during run time.
As the instance type to which *m1* is referring to is `Endangered` therefore `show()` method of `Endangered` class is executed.

BSCCS2005: Graded Assignment with Solutions
Week 4

1. Consider the code given below. [MCQ:2 points]

```
public interface Printable{
    public default void print() {
        System.out.println("not implemented");
    }
}
public abstract class Collection{
    public void print() {
        System.out.println("no element");
    }
}
public class Queue extends Collection implements Printable{
    public void print() {
        super.print();
        System.out.println("print the queue");
    }
}
public class FClass{
    public static void main(String[] args) {
        Queue q = new Queue();
        q.print();
    }
}
```

What will be the output?

- ◯ `no element`
  `not implemented`
  `print the queue`
- ◯ `not implemented`
  `print the queue`
- √ `no element`
  `print the queue`
- ◯ It generates compiler error

Solution: Whenever there is a conflict between a default method of an interface and a method in a class to be inherited, method inherited from the class gets the priority.

2. Consider the code given below. [MCQ:2 points]

```
public interface Shape{
    public double area();
    public default double volume() {
        return -1.0;
    }
}

public interface Printable{
    public default void print() {
        System.out.println("not implemented");
    }
}

public class Rectangle implements Shape, Printable{
    private double w, h;
    public Rectangle(double w_, double h_) {
        w = w_;
        h = h_;
    }
    public double area() {
        return w * h;
    }
    public void print() {
        System.out.print(area() + " ");
        System.out.print(volume());
    }
}

public class FClass{
    public static void main(String[] args) {
        Shape s = new Rectangle(20.0, 50.0);
        s.print();
    }
}
```

What will be the output?

- ◯ `1000.0` followed by a runtime error
- ◯ `not implemented`
- ◯ `1000.0 -1.0`
- √ It generates a compiler error

Solution: Since the static type of `s` is `Shape`, and `print()` method does not belong to `Shape`, the call `s.print()` is illegal.

3. Consider the code given below. [MSQ:2 points]

```
public interface NumberType{
    public double norm();
    public static void print(double x) {
        System.out.println("norm is : " + x);
    }
}

public class ComplexNum implements NumberType{
    private int r, i;
    public ComplexNum(int r_, int i_) {
        r = r_;
        i = i_;
    }
    public double norm() {
        return Math.sqrt(r*r + i*i);
    }
}

public class FClass{
    public static void main(String[] args) {
        ComplexNum c = new ComplexNum(3, 4);
        ___________________________________    //LINE1
    }
}
```

Identify the appropriate option(s) to fill in the blank at `LINE1`, such the output is:
`norm is :  5.0`

- ◯ `print(c.norm());`
- ◯ `c.print(c.norm());`
- √ `NumberType.print(c.norm());`
- ◯ `ComplexNum.print(c.norm());`

Solution: A `static` method belongs to an interface, must be invoked with the same interface name.

4. Consider the code given below. [MSQ:2 points]

```
public interface DBIF{
    public void executeStatement(String qry);
}

public class Database{
    private ConnectionObj cobj = null;
    ---------------------------------- {
        if(isValidate(u, p))
            cobj = new ConnectionObj();
        return cobj;
    }
    public boolean isValidate(String u, String p) {
        //assume the vaidation is always true
        return true;
    }
    private class ConnectionObj implements DBIF{
        public void executeStatement(String qry) {
            System.out.println("Execute: " + qry);
        }
    }
}

public class FClass{
    public static void main(String[] args) {
        DBIF con = new Database().connectDB("test", "test");
        con.executeStatement("fetch students");
    }
}
```

Identify the appropriate option to fill in blank at `LINE1` such that the output becomes `Execute:  fetch students`

- √ `public DBIF connectDB(String u, String p)`
- √ `public ConnectionObj connectDB(String u, String p)`
- ◯ `public Database connectDB(String u, String p)`
- ◯ `public static ConnectionObj connectDB(String u, String p)`

Solution: Since the method `connectDB()` is called with reference to an object of `Database`, `connectDB()` must be a non-static method.
Since the object returned by `connectDB()` need to be referred by `DBIF`, thus it should return either `DBIF` or `ConnectionObj`.

5. Consider the code given below. [MCQ:2 points]

```
public interface Inter{
  public abstract void greet();
}
public class Greetings{
  private String country;
  public void setCountry(String s){
    this.country = s;
  }
  public String getCountry(){
    return country;
  }
  public Inter checkCountry(){
    if(getCountry() == "India"){
      return new IndiaGreetings();
    }
    return new WorldGreetings();
  }
  private class IndiaGreetings implements Inter{
    public void greet(){
      System.out.println("Hello"+ " "+"India");
    }
  }
  private class WorldGreetings implements Inter{
    public void greet(){
      System.out.println("Hello"+ " "+"World");
    }
  }
}
public class Example {
  public static void main(String[] args) {
    Greetings g = new Greetings();
    g.setCountry("India");
    --------- Line 1 ---------

  }
}
```

Identify the appropriate option to fill in the blank at `Line 1` to print `Hello India` as the output.

- ◯ `g.greet();`
- √ `g.checkCountry().greet();`

- ◯ `g.IndiaGreetings.greet();`
- ◯ `g.Inter.IndiaGreetings.greet();`

Solution: To print `Hello India` greet() must be called. `IndiaGreetings` class not visible outside of `Greetings` class, but the interface `Inter` is visible which has greet() abstract method. checkCountry() is called which returns `India Greetings` object. And the greet() is called using that object.

6. Consider the code given below.

```
public class Language{
    public void show(){
        System.out.println("In Language class");
    }
    public class Programming{
        public void show(){
            System.out.println("In Programming class");
        }
    }
}
public class Example {
    public static void main(String[] args) {
        -----------Line 1-----------
    }
}
```

Identify the appropriate option to fill in the blank at `Line 1` to print `In Programming class` as the output.

- ◯ `Language.Programming.show();`
- ◯ `new Language().Programming.show();`
- √ `new Language().new Programming().show();`
- ◯ `Language.new Programming().show();`

7. Consider the Java program given below and predict the output.

[MCQ : 2 points]

```
public interface Animal{
    public void sound();
    default void eat(String animal){
        System.out.println(animal+" eats every day");
    }
}
public class Cat implements Animal{
    public void sound(){
        System.out.println("Cat meows");
    }
}
public class Dog implements Animal{
    public void sound(){
        System.out.println("Dog barks");
    }
}
public class Example{
    public static void main(String[] args){
        Animal oa1=new Cat();
        oa1.sound();
        oa1.eat("Cat");
        Animal oa2=new Dog();
        oa2.sound();
        oa2.eat("Dog");
    }
}
```

- ◯ This code generates a compilation error because an interface can't have fully implemented methods.
- ◯ Cat meows
  Dog barks
- √ Cat meows
  Cat eats every day
  Dog barks
  Dog eats every day
- ◯ Cat eats every day
  Dog eats every day

Solution: Option 3
From JDK 1.8 onwards, the interface supports default methods. Program executes successfully and all methods called.

8. Consider the Java program given below and select the correct option from among the given choices. [MCQ : 2 points]

```
public interface One{
    public void print();
}
public interface Two extends One{
    public void display();
}
public class Three implements Two{
    public void display(){
        System.out.println("This is display");
    }
}
public class  Example{
    public static void main(String[] args){
        Three three=new Three();
        three.display();
    }
}
```

- √ Compilation fails because `print()` method has not been overridden in class Three.
- ◯ Compilation fails because one interface cannot extend another interface.
- ◯ Generates no output.
- ◯ This code generates the output: `This is display`

Solution: class Three implements interface Two, interface Two extends from interface One.
class Three should override both `print()` and `display()`.

9. Choose all the correct option(s) regarding the code given below.

```
public class University{
    public int univ_id;

    private class College{
        private String college_name;

        public void getName(){
            System.out.println(college_name);
        }

        public College(String name){
            college_name = name;
        }
    }

    //As the class is private so we are using a public method
    //to return an object of the the inner class

    public College getReference(){
        return new College("IITMadras");
    }

}
public class FClass{
    public static void main(String[] args) {
        University uni = new University();
        (uni.getReference()).getName() ;

    }
}
```

[MSQ:2 points]

- ◯ This code compiles successfully and generates output: IITMadras
- ◯ This code generates a compilation error because `college_name` is a private variable.
- ◯ This code generates a compilation error because `getReference()` is actually not returning a valid object of `College` type.
- √ This code generates a compilation error because the method `getName()` is not overridden and hence not accessible.

Solution:
Method `getName` is a member function of the nested class `College` which is private, hence the only way we can access this method is by applying dynamic dispatch using method overriding. Therefore we have to use either an abstract class or interface which the inner class would inherit/implement and then we can access the `getName` method as an overridden implementation.
So correct choice would be option 4.

The Java code below models a certain functionality of a ropeway system, carrying tourists, that allows 6 travelers in each cabin. The travelers are let in one after the other into a cabin until the maximum capacity is reached (class CabinCounter). Then, the cabin is allowed to move on, and the next cabin is readied for boarding. A counter (class MasterCounter) is used to track the total number of travelers using the ropeway in a day. Based on the above information and the following code, answer questions 10 and 11.

```
public interface Counter{
    public abstract void inform();
}
public class CabinCounter{
    private MasterCounter master;

    public CabinCounter(MasterCounter mc){
        master = mc;
    }
    public void performCount(){
        int currentCabinCount = 0;
        while (currentCabinCount < 7) {
            Traveler t = new Traveler();
            if t.hasValidTicket(){
                //Let the traveler get inside the current cabin
               currentCabinCount++;
            }
        }
        //let the cabin start moving
        ..................... //Blank Line 1
    }
}
public class MasterCounter implements Counter{
    static int dayCount = 0;
    private CabinCounter cc;

    public MasterCounter(){
        cc = new CabinCounter(this); //A new cabin arrives
        cc.performCount();
    }
    public void inform(){
        this.incrementDailyCounter();
        cc = new CabinCounter(this); //A new cabin arrives
        ..................... //Blank Line 2
    }
    public void incrementDailyCounter(){
```

```
        dayCount = dayCount + 6;
    }

    public static void main(String args[]) {
        MasterCounter masterCounter = new MasterCounter();
    }
}
public class Traveler{
    ....
    public boolean hasValidTicket(){
        //This method checks if the traveler has a valid ticket.
    }
}
```

10. Which should be the line at `Blank Line 1` so that the MasterCounter is notified of a cabin becoming full?

[MCQ: 2points]

- ◯ `mc.inform();`
- √ `master.inform();`
- ◯ `cc.incrementDailyCounter();`
- ◯ MasterCounter cannot be notified because `master` is a private object inside CabinCounter.

11. Which should be the line at `Blank Line 2` so that the CabinCounter will start checking for the new cabin becoming full?

[MCQ: 2points]

- ◯ There is no need to add any line because the constructor of CabinCounter will initiate the counting for the new cabin.
- √ `cc.performCount();`
- ◯ `master.incrementDailyCounter();`
- ◯ `cc.inform();`

BSCCS2005: Graded Assignment with Solutions
Week 5

1. Consider the following code. [MSQ:2points]

```
public class Dict<K, V> {
    private K key;
    private V value;

    public Dict(K key, V value) {
        this.key = key;
        this.value = value;
    }
}

public class Example {
  public static void main(String[] args) {
    ----------Line 1--------------
  }
}
```

Which of the following statements, if placed at `Line 1`, would create an object of Dict class?

- ◯ `new Dict<Integer, Float>(1, 30);`
- ◯ `new Dict<String, int>("John", 18);`
- √ `Dict<String, Integer> d = new Dict<String, Integer>("John", 18);`
- √ `new Dict<Integer, Integer>(1, 30);`

Solution: `K, V` are type parameters of generic types. `K, V` cannot be primitive types, so option b is incorrect.
In option 1 `V` is of type Float, but is initialized with int value which is not compatible here w.r.t generics.

2. Consider the code given below and choose the correct option.

[MCQ : 2 points]

```
public abstract class Vehicle{
    abstract void capacity();
}
public class Bike extends Vehicle{
    private String count="Bike capacity at most 2 persons";
    public void capacity() {
        System.out.println(count);
    }
}
public class Auto extends Vehicle{
    private String count="Auto capacity at most 4 persons";
    public void capacity() {
        System.out.println(count);
    }
}
public class Capacity {
    public <T extends Vehicle> void seating(T obj) {
        obj.capacity();
    }
    public static void main(String[] args) {
        Capacity bike=new Capacity();
        bike.seating(new Bike());
        Capacity auto=new Capacity();
        auto.seating(new Auto());
    }
}
```

- √ This program generates output:
  `Bike capacity at most 2 persons`
  `Auto capacity at most 4 persons`
- ◯ Compilation error at: `bike.seating(new Bike());`
- ◯ Compilation error at: `auto.seating(new Auto());`
- ◯ Compilation error at: `public <T extends Vehicle> void seating(T obj)`

Solution: Type parameter T extends from `Vehicle` class.
The type parameter can accept any subclass of `Vehicle`.

3. Consider the code given below. [MCQ : 2 points]

```
public class Player{
    private  String name;
    private  String type;
    public String getName() {
        return name;
    }
    public String getType() {
        return type;
    }
    public Player(String name, String type) {
        this.name = name;
        this.type = name;
    }
    public String toString() {
        return "Player [name=" + name + ", type=" + type + "]";
    }
}
public class Captain extends Player{
    public Captain(String name, String type) {
        super(name, type);
    }
    public String toString() {
        return "Captain [name=" + getName() + ", type=" + getType() + "]";
    }
}
public class CopyArrayObjects {
    public static _____________ void copy (S[] src, T[] tgt){    //LINE1
        int i,limit;
        limit = Math.min(src.length, tgt.length);
        for (i = 0; i < limit; i++){
            tgt[i] = src[i];
        }
    }
}
public class FClass{
    public static void main(String[] args) {
        Captain captain1=new Captain("Virat", "Batting");
        Captain captain2=new Captain("Hardik", "All Rounder");
        Captain captain3=new Captain("Jasprit", "Bowling");
        Captain captain[]= {captain1, captain2, captain3};
        Player[] player= new Captain[2];
        CopyArrayObjects.copy(captain,player);
```

```
        for (int i = 0; i < player.length; i++) {
            System.out.println(player[i]);
        }
    }
}
```

Identify the correct generic type parameter at `LINE 1` such that the given code prints the below text: Identify the correct generic type parameter at `LINE 1` such that the given code prints the below text:

```
Captain [name=Virat, type=Batting]
Captain [name=Hardik, type=All Rounder]
```

- ◯ `<S, T>`
- ◯ `<T super S, S>`
- ◯ `<S, T extends S>`
- √ `<S extends T, T>`

Solution: Since the source array type (`S`) must extend target array type (`T`), option-4 is correct. Please note that `super` keyword is applicable only for the wildcard arguments.

4. Consider the code given below. [MCQ:2 points]

```
public class NumData{
    private Number n;
    public NumData(Number n) {
        this.n = n;
    }
    public String getMetaInfo() {
        if (n instanceof Integer) {
            return "Integer type, value = " + n;
        }
        else if(n instanceof Double) {
            return "Double type, value = " + n;
        }
        else if(n instanceof Character) {
            return "Character type, value = " + n;
        }
        else
            return "Number type, value = " + n;
    }
}

public class FClass{
    public static void main(String[] args) {
        Integer i0 = 10;
        Float f0 = 3.14f;
        Character c0 = 'A';
        NumData o1 = new NumData(i0);
        NumData o2 = new NumData(f0);
        NumData o3 = new NumData(c0);
        System.out.println(o1.getMetaInfo());
        System.out.println(o2.getMetaInfo());
        System.out.println(o3.getMetaInfo());
    }
}
```

Choose the correct option regarding the given code.

- ◯ This program generates output:

  ```
  Integer type, value = 10
  Number type, value = 3.14
  Character type, value = A
  ```

- ◯ This program generates output:

  ```
  Integer type, value = 10
  ```

```
Double type, value = 3.14
Character type, value = A
```

◯ This program generates output:

```
Integer type, value = 10
Number type, value = 3.14
Number type, value = A
```

√ This code generates compile time error because `Number` is not a super type of `Character`

Solution: All wrapper classes other than `Boolean`, `Character` extend the class `Number`.

5. Consider the code given below. [MCQ:2 points]

```
public interface Iterator{
    public boolean has_next();
    public Object get_next();
}
public class NumList<T extends Number> implements Iterator{
    private T[] list;
    private int idx;
    public NumList(T[] list) {
        this.list = list;
        idx = 0;
    }
    public boolean has_next() {
        if(idx < list.length - 1)
            return true;
        return false;
    }
    public Object get_next() {
        idx++;
        return list[idx];
    }
}

public class FClass{
    ____________________________________________     //LINE 1: function-header
    {
        double total = 0;
        while(lOb.has_next()) {
            total += ((Number)lOb.get_next()).doubleValue();
        }
        return total;
    }
    public static void main(String[] args) {
        Integer[] i_arr = {10, 20, 30, 40, 59};
        Double[] d_arr = {3.44, 2.65, 6.44, 1.3, 6.78};
        NumList<Integer> i_list = new NumList<Integer>(i_arr);
        NumList<Double> d_list = new NumList<Double>(d_arr);
        System.out.println(sum(i_list) + ", " + sum(d_list));
    }
}
```

Identify the appropriate function header for function `sum`, such that the output is `149.0, 17.17`

√ `public static double sum(NumList<? extends Number> lOb)`

◯ `public static double sum(NumList<Number> lOb)`

√ `public static <T extends Number> double sum(NumList<T> lOb)`

◯ `public static double sum(NumList<? super Number> lOb)`

Solution: In option-1, the parameter type `NumList<? extends Number>` is compatible with both `NumList<Integer>` and `NumList<Double>`. So, it is a correct option.
In option-2, the parameter type `NumList<Number>` is not compatible with `NumList<Integer>` and `NumList<Double>`. So, it is a wrong option.
In option-3, the parameter type `NumList<T>`, where the quntifier `T` is defined as `<T extends Number>`, is compatible with `NumList<Integer>` and `NumList<Double>`. Although, `T` is never used, it is a correct option.
In option-4, the parameter type `NumList<? super Number>` is not compatible with `NumList<Integer>` and `NumList<Double>` (`Integer` and `Double` are subtype of `Number`, not supertype). So, it is a wrong option.

6. Consider the code given below. [MSQ:2 points]

```
public class Employee{
    private String name;
    private double salary;
    public Employee(String name, double salary){
        this.name = name;
        this.salary = salary;
    }
    public String getName() {
        return name;
    }
    public double getSalary() {
        return salary;
    }
}
public class Developer extends Employee{
    //implementation with some new instance variable and methods
}
public class Manager extends Employee{
    //implementation with some new instance variable and methods
}
public class SalaryStat<T extends Employee>{
    private T[] eps;
    public SalaryStat(T[] eps) {
        this.eps = eps;
    }
    private double getTotalSalary() {
        double total = 0;
        for(int i = 0; i < eps.length; i++)
            total += eps[i].getSalary();
        return total;
    }
    public boolean greaterSalary(_______________) {    //LINE 1
        if (this.getTotalSalary() > d.getTotalSalary())
            return true;
        return false;
    }
}
public class FClass{
    public static void main(String[] args) {
        Developer[] dA = {new Developer("A", 50000.0), new Developer("B", 40000.0),
                new Developer("C", 45000.0)};
        Manager[] mA = {new Manager("X", 65000.0), new Manager("Y", 51000.0)};
```

```
        SalaryStat<Developer> dO = new SalaryStat<Developer>(dA);
        SalaryStat<Manager> mO = new SalaryStat<Manager>(mA);
        if(mO.greaterSalary(dO))
            System.out.println("managers have higher salary expenditure");
        else
            System.out.println("developers have higher salary expenditure");
    }
}
```

Identify the appropriate argument for function `greaterSalary`, such that the output is `developers have higher salary expenditure`

- ◯ `SalaryStat<T> d`
- √ `SalaryStat<?> d`
- ◯ `T d`
- √ `SalaryStat<? extends Employee> d`

Solution: Since the function `greaterSalary` compares the total salary of subclasses of `Employee`, the `LINE 1` either can be
`SalaryStatistic<? extends Employee>)`
or
`SalaryStatistic<?> d`, which matches any `SalaryStatistic` object.
For option 2 and 3, quantifier T is not defined.

Consider the class `SampleClass` in the Java code given below, and answer the questions 7 and 8.

```
import java.lang.reflect.*;
public class SampleClass{
    private final int pr_data = 9;
    private String pr_str;
    public static int pu_data;
    private SampleClass() {
        //some code
    }
    public SampleClass(int pr_data_, String pr_str_) {
        pr_str = pr_str_;
    }
    public SampleClass(SampleClass tObj) {
        this.pr_str = tObj.pr_str;
    }
    private boolean isValid() {
        //some code
        return true;
    }
    public int get_pr_data() {
        return pr_data;
    }
    public String get_pr_str() {
        return pr_str;
    }
}
```

7. What should be the statement in `Line 1` so that `Line 2` prints the number of all the public constructors in `SampleClass`?

[MCQ:2 points]

```
public class FClass{
    public static void main(String[] args) {
        Class c = Class.forName("SampleClass");
        ________________________________________ //Line 1
        System.out.println(my_const.length);     //Line 2
    }
}
```

Choose the correct option from below.

- ◯ `Constructor[] my_const = c.getMethods();`
- ◯ `Constructors my_const = c.getDeclaredConstructors();`
- √ `Constructor[] my_const = c.getConstructors();`
- ◯ `Constructor[] my_const = c.getDeclaredConstructors();`
- ◯ `Constructor my_const[] = c.getAllConstructors();`

Solution: The solution follows from the syntax of the method in the class Class to obtain the public constructors of a given class.

8. Consider the code given below. [MCQ:2 points]

```
public class FClass{
    public static void main(String[] args) {
        Class c = Class.forName("SampleClass");
        Field[] fields1 = c.getFields();
        Field[] fields2 = c.getDeclaredFields();
        for(Field f : fields1) {
            System.out.print(f.getName() + " : ");
            System.out.print(f.getType());
            System.out.println();
            System.out.println("Modifier: " +
                    Modifier.toString(f.getModifiers()));
        }
        for(Field f : fields2) {
            System.out.print(f.getName() + " : ");
            System.out.print(f.getType());
            System.out.println();
            System.out.println("Modifier: " +
                    Modifier.toString(f.getModifiers()));
        }
    }
}
```

What will the output be?

- ◯ pr_data : int
  Modifier: private final
  pr_str : class java.lang.String
  Modifier: private
  pu_data : int
  Modifier: public static
- √ pu_data : int
  Modifier: public static
  pr_data : int
  Modifier: private final
  pr_str : class java.lang.String
  Modifier: private
  pu_data : int
  Modifier: public static
- ◯ pr_data : int
  Modifier: private final
  pr_str : class java.lang.String
  Modifier: private

```
pu_data :  int
Modifier:  public static
pu_data :  int
Modifier:  public static
```

◯
```
pu_data :  int
Modifier:  public static
```

Solution: The first `for` loop prints all the public instance variables in the given class. The second `for` loop prints the public and private instance variables of the class. Only options 2 and 3 are printing both. But in option 3, the order is not correct. Hence the right answer is option 2.

9. Consider the following code. [MCQ:2points]

```
public class Algorithm{
  public <T extends Integer> boolean findOdd(T a){
    if(a % 2 == 0){
      return false;
    }
    return true;
  }
  public <T> void display(T[] arr){
    for(T i: arr){
      System.out.println(i);
    }
  }
}
```

What is class `Algorithm` converted to after type erasure?

◯
```
public class Algorithm{
  public boolean findOdd(Number a){
    if(a % 2 == 0){
      return false;
    }
    return true;
  }
  public void display(Object[] arr){
    for(Object i: arr){
      System.out.println(i);
    }
  }
}
```

◯
```
public class Algorithm{
  public boolean findOdd(Object a){
    if(a % 2 == 0){
      return false;
    }
    return true;
  }
  public void display(Object[] arr){
    for(Object i: arr){
      System.out.println(i);
    }
  }
}
```

√
```
public class Algorithm{
  public boolean findOdd(Integer a){
    if(a % 2 == 0){
      return false;
    }
    return true;
  }
  public void display(Object[] arr){
    for(Object i: arr){
      System.out.println(i);
    }
  }
}
```

◯
```
public class Algorithm{
  public boolean findOdd(T a){
    if(a % 2 == 0){
      return false;
    }
    return true;
  }
  public void display(T[] arr){
    for(T i: arr){
      System.out.println(i);
    }
  }
}
```

Solution: Type erasure replaces all type parameters in generic types with their bounds(super class) or Object if the type parameters are unbounded.

10. Consider the following code and choose the correct option regarding the same.

[MCQ:2points]

```
public interface Walkable{
    default void showPaceLength() {
        System.out.println("Average pace length : 0.4 meters");
    }
}
public class Human implements Walkable{
    double pace_length = 0.85;
    public void showPaceLength() {
        System.out.format("Average pace length : %f meters",pace_length);
    }
}

public class Mammal<T>{
    public String name;
    public T group;
    public Mammal(T obj){
        name = obj.getClass().getSimpleName();
        group = obj;
    }
    public void print() {
        System.out.println(name);
        group.showPaceLength();
    }
}
public class Test1 {
    public static void main(String[] args) {
        Mammal<Human> m = new Mammal<Human>(new Human());
        m.print();
    }
}
```

- ◯ This code generates output: Human
  Average pace length : 0.850000 meters
- ◯ This code generates output: Mammal
  Average pace length : 0.850000 meters
- ◯ This code generates a compilation error because Human type can't be passed to the generic Mammal class.
- √ This code generates a compilation error because the method `showPaceLength` could not be resolved.

◯ This code generates a compilation error because the method `getClass()` could not be resolved.

BSCCS2005: Graded Assignment with Solutions
Week 6

1. Consider the code segment given below. [MCQ:2 points]

```
__________________________________________________;    //LINE 1
names.put(40, "four");
names.put(10, "one");
names.put(30, "three");
names.put(20, "two");
System.out.println(names);
```

Identify the output for the following given cases:

1. LINE 1 is `Map<Integer, String> names = new LinkedHashMap<Integer, String>();`
2. LINE 1 is `Map<Integer, String> names = new TreeMap<Integer, String>();`
3. LINE 1 is `Map<Integer, String> names = new HashMap<Integer, String>();`

○ In case 1: prints all four key-value pairs; however, the order cannot be determined
In case 2: prints {10=one, 20=two, 30=three, 40=four}
In case 3: prints {10=one, 20=two, 30=three, 40=four}

○ In case 1: prints {10=one, 20=two, 30=three, 40=four}
In case 2: prints {10=one, 20=two, 30=three, 40=four}
In case 3: prints {10=one, 20=two, 30=three, 40=four}

○ In case 1: prints {40=four, 10=one, 30=three, 20=two}
In case 2: prints {10=one, 20=two, 30=three, 40=four}
In case 3: prints {40=four, 10=one, 30=three, 20=two}

√ In case 1: prints {40=four, 10=one, 30=three, 20=two}
In case 2: prints {10=one, 20=two, 30=three, 40=four}
In case 3: prints all four key-value pairs; however, the order cannot be determined

Solution:

- For `HashMap`, the ordering of the keys can not be determined.
- For `TreeMap`, keys are ordered by their insertion order.
- For `LinkedHashMap`, keys are in sorted order.

2. Consider the code given below. [MCQ:2 points]

```
import java.util.*;

public class FClass{
    public static void main(String[] args){
        Map<Character, Integer> frequencyTab
                                = new LinkedHashMap<Character, Integer>();
        String str = "incomprehensibilities";
        for(int i = 0; i < str.length(); i++) {
            Character c = str.charAt(i);
            ------------------------------------------
        }
        for(Map.Entry<Character, Integer> e: frequencyTab.entrySet()) {
            System.out.print("[" + e.getKey() + ", " + e.getValue() + "] ");
        }
    }
}
```

Identify the appropriate option to fill in the blank at `LINE 1`, such that the output is: `[i, 5] [n, 2] [c, 1] [o, 1] [m, 1] [p, 1] [r, 1] [e, 3] [h, 1] [s, 2] [b, 1] [l, 1] [t, 1]`

- ◯ `frequencyTab.put(c, frequencyTab.get(c) + 1);`
- ◯ `frequencyTab.putIfAbsent(c, frequencyTab.get(c) + 1);`
- √ `frequencyTab.put(c, frequencyTab.getOrDefault(c, 0) + 1);`
- ◯ `frequencyTab.putIfAbsent(c, 1);`

Solution: Option-3 is the correct answer for the following reason:
For each character from the `String str`, if character is not present in the the `Map frequencyTab`, add the character as key and `0` as value; else for key equals to the character, increase the value by one.

3. Consider the code given below. [MSQ:2 points]

```
import java.util.*;

public class FClass{
    public static void main(String[] args) {
        Map<Integer, Character> entries = new TreeMap<Integer, Character>();
        entries.put(30, 'b');
        entries.put(40, 'a');
        entries.put(20, 'd');
        entries.put(10, 'c');

        ________________________________________    //LINE 1
        for(Character c : values)
            System.out.print(c + " ");
    }
}
```

Identify the appropriate option(s) to fill in the blank at `LINE 1`, such that the output of the above code is
`a b c d`

- ◯ `Collection<Character> values = entries.values();`
- √ `TreeSet<Character> values = new TreeSet<Character>(entries.values());`
- √ `PriorityQueue<Character> values = new PriorityQueue<Character>(entries.values());`
- ◯ `LinkedList<Character> values = new LinkedList<Character>(entries.values());`

Solution: In `TreeMap`, the iterator will visit the key-value pairs in sorted order of the keys. However, if the values are added to a `TreeSet` or a `PriorityQueue`, the iterator will visit the elements in sorted order.

4. Consider the code given below. [MCQ:2 points]

```
import java.util.*;
public class FClass {
    public static void main(String args[]){
        ArrayList<String> sList = new ArrayList<String>();
        sList.add("A");
        sList.add("B");
        ListIterator<String> iter = sList.listIterator();
        if(iter.hasNext()){
            iter.next();
            iter.add("C");
        }
        if(iter.hasPrevious()){
            iter.previous();
            iter.add("D");
        }
        sList.add("E");
        System.out.println(sList);
    }
}
```

What will the output be?

- ◯ `[A, B, C, D, E]`
- ◯ `[A, B, D, C, E]`
- √ `[A, D, C, B, E]`
- ◯ `[A, D, B, C, E]`

Solution: The statements `sList.add("A");` `sList.add("B");` make the list as `[A, B]`.
The iterator `iter` start from the begining of the list. The statement `iter.next();` skip the first element. The statement `iter.add("C");` makes the list as `[A, C, B]`. The iterator `iter` positioned after `C` in the list. The statement `iter.previous();` skip the previous element, i.e. `C`. The statement `iter.add("C");` makes the list as `[A, D, C, B]`.
The statement `sList.add("E");`, add `E` to the end of the list, which makes the list as `[A, D, C, B, E]`.

5. Consider the code given below. [MCQ:2 points]

```
import java.util.*;
public class Employee implements Comparable{
    private String name;
    private double salary;
    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
    public String toString() {
        return "[" + name + " : " + salary + "]";
    }
    public int compareTo(Object e) {
        Employee d = (Employee)e;
        if(salary == d.salary)
            return name.compareTo(d.name);
        else {
            if (d.salary > salary)
                return 1;
            else if(d.salary < salary)
                return -1;
            else
                return 0;
        }
    }
}
public class FClass{
    public static void main(String[] args) {
        TreeSet<Employee> empList = new TreeSet<Employee>();
        empList.add(new Employee("raj", 30000.00));
        empList.add(new Employee("akash", 60000.00));
        empList.add(new Employee("biraj", 60000.00));
        empList.add(new Employee("vinay", 40000.00));
        for(Employee e : empList)
            System.out.println(e);
    }
}
```

What will the output be?

◯ [raj : 30000.0]
[vinay : 40000.0]
[akash : 60000.0]
[biraj : 60000.0]

○ [akash : 60000.0]
[biraj : 60000.0]
[raj : 30000.0]
[vinay : 40000.0]

√ [akash : 60000.0]
[biraj : 60000.0]
[vinay : 40000.0]
[raj : 30000.0]

○ [raj : 30000.0]
[vinay : 40000.0]
[biraj : 60000.0]
[akash : 60000.0]

Solution: As per the `compareTo()` method implementation in `class Employee`, the `Employee` objects are first sorted by the `salary` in descending order. When the there is a equality on the `salary`, those `Employee` objects are sorted by the `name` in the ascending order.

6. Consider the code given below. `HashMap m` maps each element of `ArrayList list` to its frequency of occurrence in `list`. [MSQ:2 points]

```
import java.util.*;
public class Example {
  public static void main(String[] args) {
    List<Integer> list = new ArrayList<Integer>();
    list.add(1);
    list.add(1);
    list.add(2);
    list.add(3);
    HashMap<Integer, Integer> m = new HashMap<Integer, Integer>();
    -----------SEGMENT 1-------------------------
    for (HashMap.Entry<Integer, Integer> entry : m.entrySet()){
            System.out.println("Key = " + entry.getKey() +
                                ", Value = " + entry.getValue());
    }
  }
}
```

Identify the appropriate option to fill in the blank at `SEGMENT 1`, such that the output is:

```
Key = 1, Value = 2
Key = 2, Value = 1
Key = 3, Value = 1
```

◯
```
for(Integer i : list){
        m.merge(i, 0, m.get(i)+1);
}
```

◯
```
for(Integer i : list){
        m.put(i, m.get(i)+1);
}
```

√
```
for(Integer i : list){
        m.put(i, m.getOrDefault(i,0)+1);
}
```

√
```
for(Integer i : list){
        if(m.containsKey(i)){
          m.put(i, m.get(i)+1);
        }
        else{
          m.put(i, 1);
        }
}
```

Solution: For each element from the `ArrayList list`, if element is not present as a key in the `HashMap m`, add the element as key and 0 as its value; else for key equals to the element, increase the value by one.

7. In a temple, there are two types of tickets for devotees waiting for Darshan - free tickets and tickets costing ₹100. The devotees carrying ₹100 tickets are given preference over devotees carrying free tickets. A batch of 5 devotees are let into a special queue, from which they are let inside the temple based on the ticket that they carry. After this, another batch is let in and so on. Consider the Java code that models the sequencing process in the special queue, and answer the question that follows.

```
import java.util.*;

class Devotee implements Comparable{
    String name;
    int ticket_type;

    Devotee(String p_name, int p_type){
       name = p_name;
       ticket_type = p_type;
    }
    public int compareTo(Object a) {
        Devotee d = (Devotee)a;
        if(ticket_type < d.ticket_type)
            return 1;
        else if (ticket_type > d.ticket_type)
            return -1;
        else return 0;
    }
}
public class SpecialQueue{
    public static void main(String[] args){
        Devotee[] dev_arr = new Devotee[]{new Devotee("Pavya",0),
                            new Devotee("Arya",100),new Devotee("Sana",0),
                            new Devotee("Meenu",0),new Devotee("Naina",100)};

        PriorityQueue<Devotee> specialQ = new PriorityQueue<Devotee>();
        ________________________ //Hidden Lines
    }
}
```

What should the Java code in the given blanks at `Hidden Lines` be, so that the code will print the names and ticket types of all devotees who carry ₹100 tickets before those with free tickets?

◯
```
for (int i = 0; i<5; i++){
    specialQ.add(dev_arr[i]);
    Devotee d = specialQ.poll();
```

```
        System.out.println(d.name + " "+ d.ticket_type);
    }
```

○
```
    for (int i = 0; i<5; i++){
        specialQ.add(dev_arr[i]);
    }
    for (int i = 0; i<5; i++){
        Devotee d = specialQ.poll();
        System.out.println(specialQ.poll().name + " "+
                    specialQ.poll().ticket_type);
    }
```

○
```
    for (int i = 0; i<5; i++){
        specialQ.add(dev_arr[i]);
        Devotee d = specialQ.poll();
        System.out.println(specialQ.poll().name + " "+
                    specialQ.poll().ticket_type);
    }
```

√
```
    for (int i = 0; i<5; i++){
        specialQ.add(dev_arr[i]);
    }
    for (int i = 0; i<5; i++){
        Devotee d = specialQ.poll();
        System.out.println(d.name + " "+ d.ticket_type);
    }
```

Solution:
In Option 1, adding and polling takes place in the same loop. The poll method returns what is currently the max among the ones in the heap, and hence in this case may not yield the right answer.
In Option 2, specialQ.poll() is invoked twice in the sentence. This will generate a NullPointerException because every time the poll() method is invoked, it returns and removes the current element.
Option 3 has both the issues as in Option 1 and Option 2.
In Option 4, the current 5 devotees are added to the priority queue. Using another for loop, the devotee objects are pulled out based on the priority of the ticket type.

8. Consider the following code. [MCQ:2points]

```
import java.util.*;
public interface Account{
    default void showBalance() {
        System.out.println("Abstract Account");
    }
}
public class SavingsAccount implements Account{
    double balance;
    public SavingsAccount(double amt){
        balance = amt;
    }
    public void showBalance() {
        System.out.println("SavingsAccount balance: " + balance);
    }
}
public class CurrentAccount implements Account{
    double balance;
    public CurrentAccount(double amt){
        balance = amt;
    }
    public void showBalance() {
        System.out.println("CurrentAccount balance: " + balance);
    }
}
public class Test2{
    public static void main(String args[]) {
        ArrayList<SavingsAccount> acc1 = new ArrayList<SavingsAccount>();
        acc1.add(new SavingsAccount(10000.0));
        acc1.add(new SavingsAccount(20000.5));

        ____________________________ // LINE 1
        acc2.add(0,new CurrentAccount(50000.0));
        ____________________________ // LINE 2

        for(Account t : acc2) {
            t.showBalance();
        }
    }
}
```

If the code given above produces the output:

```
CurrentAccount balance:  50000.0
SavingsAccount balance:  20000.5
```

What should be the correct choice for LINE 1 and LINE 2?

- √ LINE 1:
  `ArrayList<Account> acc2 = new ArrayList<Account>(acc1);`
  LINE 2:
  `acc2.remove(1);`
- ◯ LINE 1:
  `ArrayList<CurrentAccount> acc2 = new ArrayList<CurrentAccount>(acc1);`
  LINE 2:
  `acc2.remove(1);`
- ◯ LINE 1:
  `ArrayList<SavingsAccount> acc2 = new ArrayList<SavingsAccount>(acc1);`
  LINE 2:
  `acc2.remove(2);`
- ◯ LINE 1:
  `ArrayList<Account> acc2 = new ArrayList<Account>(acc1);`
  LINE 2:
  `acc2.remove(2);`

Solution:
Interface Account is implemented by both the classes SavingsAccount and CurrentAccount. In order to call the respective version of the overridden method `showBalance()` the ArrayList should store `Account` type reference.

the statement `acc2.add(0,new CurrentAccount(50000.0));` adds the new object of CurrentAccount at position 0 in the acc2 List, so inorder to remove the object of SavingsAccount corresponding to balance 10000.0 we should use `acc2.remove(1)`.

9. Match the following regarding a LinkedList object O.

[MCQ : 2 points]

A.`getFirst()`	I. If the list is empty, it returns null, else it returns the first element of O.
B.`peekFirst()`	II. If the list is empty, it returns null, else it returns the last element of O.
C.`removeLast()`	III. If the list is empty, it throws `NoSuchElementException`, else it returns the first element of O.
D.`peekLast()`	IV. If the list is empty, it throws `NoSuchElementException`, else it returns the last element of O.

- ◯ A-II,B-I,C-IV,D-III.
- √ A-III,B-II,C-IV,D-II.
- ◯ A-III,B-IV,C-I,D-II.
- ◯ A-III,B-I,C-II,D-IV.

Solution: `getFirst()` and `peekFirst()` are both used to get the first element of a `LinkedList` object. The difference is that if the object is empty, `getFirst()` throws an exception, whereas `peekFirst()` returns null.
`getLast()` and `peekLast()` are both used to get the last element of a `LinkedList` object. The difference is that if the object is empty, `getLast()` throws an exception, whereas `peekLast()` returns null.

10. Consider the Java program given below, and choose the correct options for Line 1 and 2.

[MCQ : 2 points]

```
import java.util.*;
public class MapEx{
    public static void main(String[] args) {
        HashMap<String,String> map1=new HashMap<String,String>();
        map1.put("India","Delhi");
        map1.put("Srilanka","Colombo");
        map1.put("Australia","Sydney");
        //Line 1
        //Line 2
        Iterator<String> it1=keys.iterator();
        Iterator<String> it2=values.iterator();
        System.out.println("Keys are:");
        while(it1.hasNext())
           System.out.println(it1.next());
        System.out.println("values are:");
        while(it2.hasNext())
           System.out.println(it2.next());
    }
}
```

- √ Line 1: `Set<String> keys=map1.keySet();`
  Line 2: `Collection<String> values=map1.values();`
- ◯ Line 1: `Set<String> keys=map1.keySet();`
  Line 2: `Set<String> values=map1.values();`
- √ Line 1: `Collection<String> keys=map1.keySet();`
  Line 2: `Collection<String> values=map1.values();`
- ◯ Not possible to get keys and values from map object separately.

Solution: `keySet()` method return `Set` as return type, you can assign it to either `Set` or `Collection` interface, because `Set` extended from `Collection` interface. `values()` method return `Collection` as return type, you can assign to `Collection` object directly.

BSCCS2005: Graded Assignment with Solutions
Week 7

1. Consider the code given below. [ARUP:MCQ:2 points]

```
import java.util.logging.*;

public class SomeClass {
    public void logIt(){
        Logger.getGlobal().info("First message");
    }
}

public class FClass {
    public static void main(String[] args){
        SomeClass obj = new SomeClass();
        obj.logIt();
        Logger.getGlobal().log(Level.FINE, "second message");
        Logger.getGlobal().setLevel(Level.OFF);
        try {
            throw new ArithmeticException();
        }
        catch(Exception e) {
            Logger.getGlobal().log(Level.SEVERE, "third message");
        }
    }
}
```

Identify the result when the code gets executed.

- ◯ It prints nothing
- √ `<date time> SomeClass logIt`
  `INFO: First message`
- ◯ `<date time> SomeClass logIt`
  `INFO: First message`
  `<date time> FClass main`
  `FINE: second message`
- ◯ `<date time> SomeClass logIt`
  `INFO: First message`
  `<date time> FClass main`
  `SEVERE: third message`

Solution: By default, the top three levels of the logging levels are logged. Thus, `second message` is not printed.
The statement `Logger.getGlobal().setLevel(Level.OFF);` suppress all the logging. Thus, the string `third message` is not printed.

2. Consider the Java classes (each written into a different file as denoted) given below.

[ARUP:MCQ:2 points]

```
//FILE-1
package in.ac.iitm;
import java.util.logging.*;
public class SomeClass{
    private final static Logger logbook = Logger.getLogger("in.ac.iitm");
    public void doIt(){
        logbook.warning("start of doIt() in in.ac.iitm");
        logbook.setLevel(Level.OFF);
        logbook.warning("end of doIt() in in.ac.iitm");
    }
}

//FILE-2
package in.ac.iitm.onlinedegree;
import java.util.logging.*;
public class SomeClass{
    private final static Logger logbook =
                            Logger.getLogger("in.ac.iitm.onlinedegree");
    public void doIt(){
        logbook.warning("start of doIt() in in.ac.iitm.onlinedegree");
        logbook.setLevel(Level.OFF);
        logbook.warning("end of doIt() in in.ac.iitm.onlinedegree");
    }
}

//FILE-3
public class FClass{
    public static void main(String[] args){
        in.ac.iitm.SomeClass obj1 = new in.ac.iitm.SomeClass();
        in.ac.iitm.onlinedegree.SomeClass obj2 =
                                    new in.ac.iitm.onlinedegree.SomeClass();

        obj1.doIt();
        obj2.doIt();
    }
}
```

Identify the result when the code gets executed.

√ <date time> in.ac.iitm.SomeClass doIt
WARNING: start of doIt() in in.ac.iitm

- ◯ <date time> in.ac.iitm.SomeClass doIt
  WARNING: start of doIt() in in.ac.iitm
  <date time> in.ac.iitm.onlinedegree.SomeClass doIt
  WARNING: start of doIt() in in.ac.iitm.onlinedegree
- ◯ <date time> in.ac.iitm.SomeClass doIt
  WARNING: start of doIt() in in.ac.iitm
  <date time> in.ac.iitm.SomeClass doIt
  WARNING: end of doIt() in in.ac.iitm
  <date time> in.ac.iitm.onlinedegree.SomeClass doIt
  WARNING: start of doIt() in in.ac.iitm.onlinedegree
- ◯ <date time> in.ac.iitm.SomeClass doIt
  WARNING: start of doIt() in in.ac.iitm
  <date time> in.ac.iitm.SomeClass doIt
  WARNING: end of doIt() in in.ac.iitm
  <date time> in.ac.iitm.onlinedegree.SomeClass doIt
  WARNING: start of doIt() in in.ac.iitm.onlinedegree
  <date time> in.ac.iitm.onlinedegree.SomeClass doIt
  WARNING: end of doIt() in in.ac.iitm.onlinedegree

Solution: The program logger two different logger for two different `SomeClass` classes.
Since, logger names are hierarchical, if we set log level as `OFF` on the logger `in.ac.iitm`, then the child logger `in.ac.iitm.onlinedegree` inherits that level.

3. Consider the Java code given below. [ARUP:MCQ:2 points]

```
public class MainClass{
    public static double compute(int a, int b){
        int c = 0;
        assert a > 0: "a must be > 0";    //assert-1
        assert b > 0: b;                  //assert-2
        c =  a / b;
        assert c >= 0: c;                 //assert-3
        return Math.sqrt(c);
    }
    public static void main(String[] args){
        int a = 10;
        int b = -5;
        assert b != 0: "b == 0";          //assert-4
        compute(a, b);
    }
}
```

Identify the first `assert` statement that throws the `AssertionError` when the class is executed as:
`java -ea MainClass`

- ◯ `assert-1`
- √ `assert-2`
- ◯ `assert-3`
- ◯ `assert-4`

Solution: The condition given for the assert statement `assert-2` is false, so it throws the `AssertionError`.

4. Consider the Java code given below. [ARUP:MCQ:2 points]

```
public class DOBRegistration{
    private int day, month, year;
    public DOBRegistration(int day, int month, int year){
        assert 0 < day && day <= 31: "day :" + day;        //assert-1
        this.day = day;
        assert 0 < month && month <= 12: "day :" + day;    //assert-2
        this.month = month;
        this.year = year;
    }
}
public class JobApplication{
    private int age;
    public JobApplication(int age){
        assert age >= 18: "invalid age for job";            //assert-3
        this.age = age;
    }
}
public class TaxReturn {
    private double income;
    public TaxReturn(double income){
        assert income >= 100000.00: income;                 //assert-4
        this.income = income;
    }
}
public class FClass3{
    public static void main(String[] args){
        DOBRegistration dr = new DOBRegistration(2, 23, 1879);
        JobApplication ja = new JobApplication(20);
        TaxReturn tr = new TaxReturn(75000.00);
    }
}
```

Identify the `assert` statement that throws the `AssertionError` when the class is executed as:

```
java -ea:...  -da:DOBRegistration FClass
```

- ◯ `assert-1`
- ◯ `assert-2`
- ◯ `assert-3`
- √ `assert-4`

Solution: Since assertions are enabled for all the classes exxcept `class DOBRegistration`. Thus, assert statement `assert-4` throws `AsserionError`.

5. Consider the following Java code and choose the correct option.

[Anand : MCQ : 2 points]

```
public class Example {
    public static void main(String[] args) {
        int a=10,b=0;
        try{
            int c=a/b;
            System.out.println("Quotient is "+c);
        }
        catch (Exception ae){
            System.out.println("Exception handled");
        }
        catch (ArithmeticException ae){
            System.out.println("ArithmeticException handled");
        }
    }
}
```

- ◯ This code generates the output:
  `Exception handled`
- ◯ This code generates the output:
  `ArithmeticException handled`
- ◯ This code generates the output:
  `Exception handled`
  `ArithmeticException handled`
- √ Compilation error

Solution: catch blocks must be ordered from most specific exceptions to most general exception; otherwise the specific exception block after general exception block becomes unreachable code.

6. Consider the following Java code and choose the correct option. [Anand : MCQ : 2 points]

```
public class Example{
    public static void main(String[] args) {
        try{
            int a=10/0;
        }
        finally{
            System.out.println("In finally block");
        }
        System.out.println("Program execution finished");
    }
}
```

- ◯ Compilation error
- √ This program terminates abnormally after printing the message:
  `In finally block`
- ◯ The program terminates successfully after printing the message:
  `In finally block`
- ◯ The program terminates successfully after printing the message:
  `In finally block`
  `Program execution finished`

Solution: In the above program, there is no corresponding catch block for handling `ArithmeticException`, hence the program terminates abnormally.

7. Consider the following Java code and choose the correct option for `Line 1` such that the code prints: `String index out of its range`. [Anand : MCQ : 2 points]

```
public class Example {
    public static void main(String[] args) {
        String name = "IIT Madras";
        try{
            System.out.println(name.charAt(10));
        }
        //Line 1
    }
}
```

- ◯ `catch (StringIndexOutOfBoundsException e){ System.out.println("String index out of its range"); }`
- ◯ `catch (Exception e){ System.out.println("String index out of its range"); }`
- ◯ `catch (Throwable t){ System.out.println("String index out of its range"); }`
- √ All of the above

Solution: In the above program, the statement `name.charAt(10)` throws `StringIndexOutOfBoundsException` which can be caught by using StringIndexOutOfBoundsException/Exception/Thowable catch blocks.

8. Consider the following Java code and choose the correct option.

[Anand : MCQ : 2 points]

```
public class Example{
    public void show(){
        NullPointerException e = new NullPointerException();
        e.initCause(new ArithmeticException());
        throw e;
    }
    public static void main(String[] args) {
        Example object = new Example();
        try{
            object.show();
        }
        catch (Exception e){
            System.out.println(e);
            System.out.println(e.getCause());
        }
    }
}
```

- √ This program generates the output.
  `java.lang.NullPointerException`
  `java.lang.ArithmeticException`
- ◯ This program generates the output.
  `java.lang.NullPointerException`
- ◯ This program generates the output.
  `java.lang.ArithmeticException`
- ◯ Compilation error

Solution: In above program used exception rethrow concept.
Here java.lang.NullPointerException chained with the java.lang.ArithmeticException

9. Consider the following Java code and choose the correct option. [Anand : MCQ : 2 points]

```
public class Example{
    public static void main(String[] args) {
        int a=10,b=0;
        try{
            int c=a/b;
            System.out.println("Quotient is "+c);
        }
        catch (ArithmeticException e){
            System.out.println(10/0);
            System.out.println("b value should not be zero");
        }
        catch (Exception e){
            System.out.println("Exception handled");
        }
    }
}
```

- ◯ Compilation error
- ◯ The program terminates normally after printing the message: `b value should not be zero`
- ◯ The program terminates normally after printing the message: `Exception handled`
- √ The program terminates abnormally due to unhandled exception(s).

Solution: The statement `int c = a/b;` causes an `ArithmeticException`, which results in the execution of the corresponding catch block. However, the statement `System.out.println(10/0);` inside the catch block throws another `ArithmeticException`, which remains unhandled. As a result, the program gets terminated abruptly.

10. Consider the following Java code and choose the correct option(s). [Anand : MSQ : 2 points]

```
//Data.java
package util.iitm.java;
public class  Data{
    void show(){
        System.out.println("This is show");
    }
}

//UseData.java
package iitm.java.program;
public class UseData {
    public static void main(String[] args) {
        new util.iitm.java.Data().show();
    }
}
```

- ◯ Compilation error in `Data.java`
- √ `Data.java` gets compiled.
- √ Compilation error in `UseData.java`
- ◯ `UseData.java` gets compiled.

Solution: `Data.java` gets compiled without any errors and `Data.class` is created under the `util.iitm.java` package.
Compilation error in `UseData.java` because `show()` is not declared as `public`, and hence you cannot access it outside the package.

11. Consider the following two source code files located in two different packages as shown.

[MSQ : 2 points]

//Adder.java

```
package iitm;
public class Adder {
    int add(int n1, int n2, int n3) {
        return n1+n2+n3;
    }
    protected int add(int n1,int n2) {
        return n1+n2;
    }
}
```

//Test1.java

```
package test;
import iitm.*;

class Calculator extends Adder{
    public void calculate() {
        System.out.println(this.add(7,8,9)); // LINE 1
        System.out.println(this.add(9,10));  // LINE 2
    }
}
public class Test1{
    public static void main(String args[]) {
        Adder a1 = new Adder();

        System.out.println(a1.add(4,5));   // LINE 3
        System.out.println(a1.add(1,2,3)); // LINE 4

        new Calculator().calculate();
    }
}
```

Choose the correct option regarding these two .java files.

- ◯ LINE 1 will not lead to compilation error.
- √ LINE 2 will not lead to compilation error.
- ◯ LINE 3 will not lead to compilation error.
- ◯ LINE 4 will not lead to compilation error.

Solution: `protected` members of a class can only be accessed in subclasses within the package as well as outside the package.
Members of a class where access specifier is not mentioned explictly are treated as ***package-private*** types and can only be accessed within that specific package. `public` members are accessible through out all packages and all classes
Therefore the `Adder` object `a1` inside the `test1` class's `main` method can only access public members of `Adder` class.
The `Calculator` class can access the `protected` members of the `Adder` class along with `public` members (if any) because `Calculator` class is a subclass of `Adder`.

BSCCS2005: Graded Assignment with Solutions
Week 8

1. Consider the code given below. [MCQ:2 points]

```
public class Point{
    private int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public void setX(int x) {
        this.x = x;
    }
    public void setY(int y) {
        this.y = y;
    }
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
    public Object clone() throws CloneNotSupportedException{
        return super.clone();
    }
}

public class FClass{
    public static void main(String[] args) {
        try {
            Point p1 = new Point(10, 20);
            Point p2 = p1;
            Point p3 = (Point)p1.clone();
            p1.setX(100);
            p1.setY(200);
            System.out.println(p1 + " , " + p2 + ", " + p3);
        }
        catch(CloneNotSupportedException e) {
            System.out.println("clone() not supported");
        }
    }
}
```

What will the output be?

- ◯ `(100, 200), (100, 200), (100, 200)`
- ◯ `(100, 200), (100, 200), (10, 20)`
- ◯ `(100, 200), (10, 20), (10, 20)`
- √ `clone() not supported`

Solution: Since class `Point` does not implement `Cloneable`, an attempt to call `clone()` would generate `CloneNotSupportedException` exception. Thus, it prints `clone() not supported`.

2. Consider the code given below. [MCQ:2 points]

```
public class Product implements Cloneable{
    private String prodname;
    private double prodprice;
    public Product(String prodname, double prodprice) {
        this.prodname = prodname;
        this.prodprice = prodprice;
    }
    public Product(Product p) {
        this.prodname = p.prodname;
        this.prodprice = p.prodprice;
    }
    public void setProdname(String prodname) {
        this.prodname = prodname;
    }
    public void setProdprice(double prodprice) {
        this.prodprice = prodprice;
    }
    public String toString() {
        return prodname + " : " + prodprice;
    }
    protected Product clone() throws CloneNotSupportedException{
        return (Product)super.clone();
    }
}

public class FClass{
    public static void main(String[] args) {
        try {
            Product p1 = new Product("Pen", 100.0);
            Product p2 = new Product(p1);
            Product p3 = p1;
            Product p4 = p1.clone();
            p1.setProdname("Pencil");
            p1.setProdprice(30.0);
            System.out.print(p1 + ", " + p2 + ", " + p3 + ", " + p4);
        }
        catch(CloneNotSupportedException e) {
            System.out.println("clone() not supported");
        }
    }
}
```

What will the output be?

- ◯ `Pencil : 30.0, Pencil : 30.0, Pencil : 30.0, Pencil : 30.0`
- ◯ `Pencil : 30.0, Pen : 100.0, Pencil : 30.0, Pencil : 30.0`
- √ `Pencil : 30.0, Pen : 100.0, Pencil : 30.0, Pen : 100.0`
- ◯ `clone() not supported`

Solution: Since `p2` allocates a new and copies the instance variables from `p1` (using copy constructor), the changes in `p1` is not reflected on `p2`.
Since, `p1` and `p3` refers to the same object, any change to `p1` would be reflected on `p3`.
However, `p4` creates a separate copy of the `p1` object. Thus, the changes in `p1` are not reflected on `p4`.

3. Consider the code given below. [MCQ:2 points]

```
public class FClass{
    public static void main(String[] args) {
        var a = 10;
        var b = "20";
        var c = a + b + 30;
        var d = a + 30 + b;
        System.out.println(c + ", " + d);
    }
}
```

What will the output be?

- ◯ `60, 60`
- ◯ `3030, 4020`
- √ `102030, 4020`
- ◯ `102030, 103020`

Solution: `a` has inferred type `int`.
`b` has inferred type `String`.
Thus, `a + b + 30 = "1020" + 30 = "102030"`,
and `a + 30 + c = 40 + "20" = "4020"`.

4. Consider the code given below. [MCQ:2 points]

```
public class Employee{
    public Employee(){}
    public String toString(){
        return "from Employee";
    }
}
public class Manager extends Employee{
    public Manager(){}
    public String toString(){
        return "from Manager";
    }
}
public class FClass{
    public static void main(String[] args) {
        Employee e = new Manager();
        var o1 = e;
        var o2 = new Employee();
        var o3 = new Manager();
        System.out.println(o1);
        System.out.println(o2);
        System.out.println(o3);
    }
}
```

What will the output be?

- ◯ from Employee
  from Employee
  from Manager
- √ from Manager
  from Employee
  from Manager
- ◯ from Manager
  from Manager
  from Manager
- ◯ from Employee
  from Employee
  from Employee

Solution: `o1` has inferred type `Manager`.
`o2` has inferred type `Employee`.
`o3` has inferred type `Manager`.

5. Consider the code given below. [MCQ:2 points]

```
public class FClass{
    public static void main(String[] args) {
        var a = 100;
        a = 10.5;    //LINE 1
        var b = 5;
        var c = a / b;
        System.out.println(c);
    }
}
```

Choose the correct option regarding the code.

- ◯ It generates output: `2.1`
- ◯ It generates output: `2`
- ◯ It generates output: `20`
- √ It generates a compiler error at `LINE 1` due to incompatible types `int` and `double`.

Solution: For the statement `var a = 100;`, the type of `a` is inferred from the initial value. So, `a` is a `int`. Thus, the compiler does not allow `a = 10.5;`.

6. The `merge` method of `Map` has three arguments - key, value and reference to a function accepting two arguments - and merges the old value with the new value for a given key. Consider the code given below. [MCQ:2 points]

```
import java.util.*;
public class FClass{
    public static void main(String[] args){
        Map<String, Integer> order1 = new TreeMap<String, Integer>();
        order1.put("Pen", 3);
        order1.put("Pencil", 10);
        order1.put("Notebook", 4);
        order1.put("Paper", 50);
        Map<String, Integer> order2 = new TreeMap<String, Integer>();
        order2.put("Pencil", 20);
        order2.put("Eraser", 5);
        order2.put("Paper", 10);
        order2.put("Pen", 7);
        Map<String, Integer> totalSell = new TreeMap<String, Integer>();

        for(Map.Entry<String, Integer> e : order1.entrySet())
            totalSell.put(e.getKey(), e.getValue());

        for(Map.Entry<String, Integer> e : order2.entrySet())
            totalSell.merge(e.getKey(), e.getValue(), (x, y) -> y + x);

        System.out.println(totalSell);
    }
}
```

Choose the correct option regarding the code.

- √ It generates output: {`Eraser=5, Notebook=4, Paper=60, Pen=10, Pencil=30`}
- ◯ It generates output: {`Eraser=5, Notebook=4, Paper=10, Pen=7, Pencil=20`}
- ◯ It generates output: {`Eraser=5, Notebook=4, Paper=50, Pen=3, Pencil=10`}
- ◯ It generates runtime exception: `NullPointerException`

Solution: For each entry in `order2`, The statement `totalSell.merge(...)`, finds out if the key is already present in the `totalSell`.If the key does exists in `totalSell`, it would be added to `totalSell` alng with corresponding value. Otherwise, it gets the old value corresponding to the key, add it with the new value, and update `totalSell`.

7. Consider the code given below. [MSQ:2 points]

```
import java.util.stream.*;
import java.util.*;
public class Product{
    private String name;
    private double price;
    public Product(String n, double p){
        name = n;
        price = p;
    }
    public double getPrice(){
        return price;
    }
    public String toString(){
        return name + " : " + price;
    }
}
public class FClass{
    public static void main(String[] args){
        var pList = new ArrayList<Product>();
        pList.add(new Product("Pen", 10.0));
        pList.add(new Product("Pencil", 5.0));
        pList.add(new Product("Notebook", 40.0));
        pList.add(new Product("Eraser", 8.0));

        var outputList = ____________________;    //LINE 1
        outputList.forEach(n -> System.out.println(n));
    }
}
```

Identiy the appropriate option(s) to fill in the blank at `LINE 1` such that the output of the program is:

```
Pen : 10.0
Notebook : 40.0
```

- √ `pList.stream().filter(x -> x.getPrice() >= 10)`
- ◯ `pList.stream().filter(x -> x >= 10)`
- √ `pList.stream().filter((Product x) -> x.getPrice() >= 10)`
- ◯ `pList.stream().takeWhile(x -> x.getPrice() >= 10)`

Solution: The given program extracts the `Product` objects from the `pList` that have `price >= 10`. In order to access `price`, the accessor function `getPrice` is required to be invoked in `filter`.

8. Consider the code given below. [MCQ:2 points]

```
import java.util.stream.*;
import java.util.*;

public class FClass{
    public static void main(String[] args){
        int m = 15;
        Stream.iterate(1, n -> n + 1)
            .limit(m)
            .filter(n -> m % n == 0)
            .forEach(n -> System.out.print(n + " "));
    }
}
```

Identify the appropriate option for the above code.

- ◯ It produces no output.
- √ It produces output as `1 3 5 15`
- ◯ It produces output as `1 2 3 4 5 6 7 8 9 10 11 12 13 14 15`
- ◯ It generates compiler error due to invalid pipeline

Solution: `iterate(1, n -> n + 1)` generate a stream $123\cdots$.
`limit(m)` limit the stream at $m = 15$.
`filter(n -> m % n == 0)` filters the values which divides m, i.e. 13515. `forEach(n -> System.out.print(n + " ")` prints each element.

9. Consider the Java code given below, and answer the question that follows.

[MCQ:2pts]

```
import java.util.stream.Stream;
import java.util.*;

public class StreamEx{
    int j=0;
    public static void main(String []args){
        ArrayList<Integer> list = new ArrayList<Integer>();

        for(int i = 1; i< 10; i++){
            list.add(i);
        }
        //CODE BLOCK 1
    }
}
```

From among the options, what should be filled in `CODE BLOCK 1` so that the code prints the even numbers between 1 and 9?

- √ `Stream<Integer> stream = list.stream().filter(j -> j%2 == 0);`
  `stream.forEach(s -> System.out.println(s));`
- ◯ `Stream<Integer> stream = list.stream().filter(j -> j%2 = 0);`
  `stream.forEach(s -> System.out.println(s));`
- √ `Stream<Integer> stream = list.stream();`
  `stream = stream.filter(j -> j%2 == 0);`
  `stream.forEach(s -> System.out.println(s));`
- ◯ `Stream<Integer> stream = list.stream();`
  `List<Integer> newList = (List)stream.filter(j -> j%2 == 0);`
  `newList.forEach(s -> System.out.println(s));`

Solution: Option 1 is correct.
In Option 2, the filtering condition must return a boolean value, here it is an assignment.
Option 3 is only a split form of Option 1, and hence is correct. In Option 4, the output of filter cannot be directly assigned to a List object (typecasting is not possible from Stream to List).

10. What is the likely outcome of the following code? [MCQ:2pts]

```
import java.util.stream.Stream;
public class StreamRandom {
    public static void main(String[] args) {
        Stream random = Stream.generate(Math::random)
                              .map(i -> Math.round(i * 100))
                              .filter(j -> j > 50).limit(5);
        random.forEach(s -> System.out.println(s));
    }
}
```

√ 92
71
98
96
52

◯ 78
39
49
29
54

◯ 53
94
86
75
82
74

◯ 0.14161383465857424
0.9951624577496674
0.1500299618014398
0.8523339358885837
0.20299930897974205

◯ 6200
7400
9100
5100
5800

Solution: `Math::round` generates random numbers between 0.0 and 1.0. `Math.round(i * 100)` multiplies it by 100 and rounds it to an integer. `filter(j`

`-> j > 50)` filters in only those numbers that are greater than 50. `limit(5)` limits the number of numbers generated to 5. Option 1 is the only option that satisfies all the conditions.

BSCCS2005: Graded Questions with Solutions
Week 9

1. Consider the following code and choose the correct option.

```
import java.util.*;
public class User{
   public String name;
   public String pass;
   User(String name,String pass){
      this.name=name;
      this.pass=pass;
   }
   String checkUser(){
      if(pass.equals("1234")){
         return name;
      }
      else{
         return pass;
      }
   }
}
public class Test{
   public static void main(String[] args){
      Optional<String> op1=Optional.ofNullable(new User(null,"1234").checkUser());
      Optional<String> op2=Optional.ofNullable(new User("Moon","4321").checkUser());
      System.out.println(op1.orElse("Sun")+"\n"+op2.orElse("Sun"));
   }
}
```

- ◯ This program generates output:
  Sun
  Sun
- ◯ This program generates output:
  1234
  Moon
- √ This program generates output:
  Sun
  4321
- ◯ This program generates output:
  Sun
  Moon

Solution: `ofNullable(T value)` method returns an `Optional` describing the specified value, if non-null, otherwise returns an empty `Optional`.
`orElse(T other)` method returns the value if present, otherwise returns `other`.

2. Consider the following Java code and choose the correct option.

```
import java.util.*;
public class Example{
   Optional<Integer> count;
   Example(Optional<Integer> count){
      this.count=count;
   }
   void show(){
      Optional<Integer> num=count.filter(n->n>18);
      if(num.isPresent()){
         System.out.println(num.get());
      }
      else{
         System.out.println(count.map(n->18-n).get());
      }
   }
}
public class Test{
   public static void main(String[] args){
      Optional<Integer> op1=Optional.of(20);
      Optional<Integer> op2=Optional.of(10);
      new Example(op1).show();
      new Example(op2).show();
   }
}
```

- ◯ This program generates NullPointerException.
- √ This program generates output:
  20
  8
- ◯ This program generates output:
  20
  10
- ◯ This program generates output:
  18
  8

Solution: `filter(Predicate<? super T> predicate)` method returns an `Optional` describing the value if a value is present and the value matches the given predicate, otherwise, returns an empty `Optional`.
`map(Function<? super T,? extends U> mapper)` method takes a mapping function as an input, which accept reference of type T and return U. If the value of T is null, `map()` returns empty `Optional`. If the mapping function is null or returns a null result, `map()` throws `NullPointerException`.

3. Consider the following Java code and choose the correct option.

```
import java.util.*;
public class UserException extends RuntimeException {
   public UserException(String str){
      super(str);
   }
}
public class Example{
   Optional<Integer> op;
   Example(Optional<Integer> op){
      this.op=op;
   }
   void show(){
      System.out.println(op.orElseThrow(()->
                new UserException("No number detected")));
   }
}
public class Test{
   public static void main(String[] args){
      Optional<Integer> op=Optional.of(100);
      Optional<Integer> op2=Optional.empty();
      new Example(op).show();
      new Example(op2).show();
   }
}
```

- √ This program prints 100 followed by UserException with message "No number detected".
- ◯ This program generates NullPointerException .
- ◯ This program generates output:
  100
  null
- ◯ This program generates output:
  100
  Optional.empty

Solution: `orElseThrow(Supplier<? extends X> exceptionSupplier)` returns the contained value, if present, otherwise throw an exception created by the provided `supplier`.

4. Consider the following Java code and choose the correct option.

```
import java.util.*;
import java.util.stream.*;

public class Test{
    public static void main(String[] args){
        var list=new ArrayList<Integer>();
        for(int i=10;i>0;i--){
            list.add(i);
        }
        IntSummaryStatistics stat = list.stream().
                collect(Collectors.summarizingInt(x->x));
        System.out.println(stat.getAverage()+"\n"+
                stat.getSum()+"\n"+stat.getMax()+"\n"+stat.getMin());
    }
}
```

- √ This program generates output:
  5.5
  55
  10
  1
- ◯ This program generates output:
  5
  55
  10
  1
- ◯ This program gives compile time error because list cannot be converted to stream.
- ◯ This program generates compile time error with message "incompatible types: possible lossy conversion from double to int".

5. Consider the code given below.

```
import java.util.*;
import java.util.stream.*;
public class StreamExample2 {
    public static void main(String[] args) {
        var fruits = new ArrayList<String>();
        fruits.add("Bananas");
        fruits.add("Apples");
        fruits.add("Kiwis");
        fruits.add("Grapes");
        fruits.add("pomegranates");
        fruits.add("watermelons");
        Stream<String> fruitstream=fruits.stream();
        IntSummaryStatistics summary =
                fruitstream.collect(Collectors.summarizingInt(String::length));
        double avgLengthFruit = summary.getAverage();
        int maxLengthFruit = summary.getMax();
        System.out.println(avgLengthFruit);
        System.out.println(maxLengthFruit);
    }
}
```

Choose the correct option regarding the code.

- √ This program generates output:
  `7.833333333333333`
  `12`
- ◯ This program generates output:
  `0`
  `12`
- ◯ This program generates a compilation error.
- ◯ This program terminates abnormally due to unhandled exception(s).

Solution: getAverage() this method will return the average string length of the all the strings in `fruits` object.
getMax() this method will return the length of the maximum length string among all the strings in `fruits` object.

6. Consider the code given below.
Assume that the file "`E:\\Files\\java.txt`" contains the following text.
`Java is a programming language.`

```
import java.io.*;
public class Example {
    public static void main(String[] args) {
        try {
            var in=new FileInputStream("E:\\Files\\java.txt");
            var din=new DataInputStream(in);
            System.out.println(din.read());
            System.out.println(din.readLine());
        }
        catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ This program generates output:
  `J`
  `Java is a programming language.`
- ◯ This program generates output:
  `74`
  `Java is a programming language.`
- √ This program generates output:
  `74`
  `ava is a programming language.`
- ◯ This program generates a compilation error.

Solution: read() methods read only one character and will return ASCII value of that character.
readLine() method reads an entire line from the file.

7. Consider the code given below.
Assume that there is no file named "`E:\\Files\\tree.txt`".

```
import java.io.*;
public class Example {
    public static void main(String[] args) {
        try {
            var out=new FileOutputStream("E:\\Files\\tree.txt");
            var dout=new DataOutputStream(out);
            String data="Trees give us fresh air.";
            dout.writeBytes(data);
            System.out.println("Data written to file successfully");
        }
        catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            System.out.println("Program execution finished.");
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ This program generates compilation error.
- ◯ This program terminates abnormally due to unhandled exception(s).
- √ This program generates output:
  `Data written to file successfully`
  `Program execution finished.`
- ◯ This program terminates abnormally after printing the message:
  `Program execution finished.`

Solution: While writing data into the file, if such a file does not exist, then it will be created and data will be written into it.

8. Consider the following Java code and choose the correct option.

```
import java.io.*;
public class Example{
    String str;
    int x;
    Example(String str, int x){
        this.str = str;
        this.x = x;
    }
    public String getStr(){
        return str;
    }
    public int getX(){
        return x;
    }
}
public class Test{
    public static void main(String[] args) throws Exception{
        ObjectOutputStream os =
                new ObjectOutputStream(new FileOutputStream("File.ser"));
        os.writeObject(new Example("Moon",1));
        FileInputStream fis=new FileInputStream("File.ser");
        ObjectInputStream ois=new ObjectInputStream(fis);
        Example e=(Example)ois.readObject();
        System.out.print(e.getStr()+"\n"+e.getX());
    }
}
```

- ◯ This program generates compile time error because class `Example` does not implement `Serializable`.
- √ This program compiles successfully but throws `NotSerializableException` at runtime.
- ◯ This program generates output:
  Moon
  1
- ◯ This program generates output:
  1
  Moon

9. Consider the following Java code and choose the correct option.

```
import java.io.*;
public class Example implements Serializable{
    transient String str="Moon";
    transient final int x=1;
}
public class Test{
    public static void main(String[] args) throws Exception{
        ObjectOutputStream os = new ObjectOutputStream(new FileOutputStream
                                ("File.ser"));
        os.writeObject(new Example());
        FileInputStream fis=new FileInputStream("File.ser");
        ObjectInputStream ois=new ObjectInputStream(fis);
        Example e=(Example)ois.readObject();
        System.out.print(e.str+"\n"+e.x);
    }
}
```

- ◯ This program generates output:
  null
  0
- √ This program generates output:
  null
  1
- ◯ This program generates output:
  0
  1
- ◯ This program compiles successfully but throws `NotSerializableException` at runtime.

Solution: Once a variable is declared `final`, at runtime it is no longer in variable form but in value form. As `final` variable participates in serialization in value form and not in variable form therefore JVM can not check for transient. Hence there is no impact of transient on `final` variable.

10. Consider the following Java code and choose the correct option.

```
import java.io.*;
import java.util.ArrayList;
public class Employee implements Serializable{
    transient String empName;
    static String company="ABC limited";
    double salary;
    public Employee(String empName, double salary) {
        this.empName = empName;
        this.salary = salary;
    }
}
public class SerialTest1{
    public static void main(String[] args) throws Exception{
        var fos=new FileOutputStream("Employee.txt");
        var os=new ObjectOutputStream(fos);
        ArrayList<Employee> list=new ArrayList<Employee>();
        list.add(new Employee("John",10000.00));
        list.add(new Employee("Jock",20000.00));
        os.writeObject(list);
        var fis=new FileInputStream("Employee.txt");
        var ois=new ObjectInputStream(fis);
        ArrayList<Employee> empList=(ArrayList<Employee>)ois.readObject();
        for(Employee emp:empList){
            System.out.print(emp.empName+" "+emp.salary+" "+emp.company);
        }
    }
}
```

- ◯ Compilation error.
- ◯ The program terminates abnormally during the deserialization process.
- ◯ The program terminates abnormally during the serialization process.
- √ This program generates output:

```
null 10000.0 ABC limited
null 20000.0 ABC limited
```

Solution:

BSCCS2005: Graded with Solutions
Week 10

1. Consider the program given below.

```
class Counter implements Runnable{
    boolean stopRequested = false;
    long count = 0;
    public void run() {
        while (!stopRequested) {
            count++;
            if (count==1000000) {
                stopRequested = true;
            }
        }
    }
    public void setStop(boolean stop){
        stopRequested = stop;
    }
    public long getCount(){
        return count;
    }
}
public class ThreadEx {
    public static void main(String[] args) throws InterruptedException {
        Counter ctr = new Counter();
        Thread backgroundThread = new Thread(ctr);
        backgroundThread.start();
        Thread.sleep(1);
        ctr.setStop(true);
        System.out.println(ctr.getCount());
    }
}
```

What will the output be?

- ◯ 0
- ◯ 1000000
- √ Some whole number between 1 and 1000000
- ◯ 999999

2. A teacher has to find the maximum marks in a subject for a class of 50 students. The teacher splits the class into two equal groups, and compute the maximum in each group simultaneously, and then find the maximum of the two. Based on this information, consider the Java code given below.

```
import java.util.*;

class SumCompute implements Runnable{
    ArrayList<Integer> half_batch = new ArrayList();
    int max;
    public SumCompute(ArrayList<Integer> hb) {
        half_batch = (ArrayList<Integer>)hb.clone();
        max = -1;
    }
    public void run() {
        max = Collections.max(half_batch);
    }
    public int getMax() {
        return max;
    }
}
public class MaxMarks {
    public static void main(String[] args) {
        int max = -1;

        //Accept the marks into
        //two ArrayList<Integer> objects batch1 and batch2

        SumCompute sc1 = new SumCompute(batch1);
        SumCompute sc2 = new SumCompute(batch2);
        Thread t1 = new Thread(sc1);
        Thread t2 = new Thread(sc2);
        t1.start();
        t2.start();
        //Line 1
        if (sc1.getMax() >= sc2.getMax()) {
            max = sc1.getMax();
        }
        else {
            max = sc2.getMax();
        }
        System.out.println(max);
    }
}
```

What should be added at `Line 1` so that the program will always give the correct result?

- ◯ Thread.sleep(1000);
- √ `while(sc1.getMax() == -1 || sc2.getMax() == -1) {}`
- ◯ `while(sc1.getMax() == -1 && sc2.getMax() == -1) {}`
- ◯ No code is required at `Line 1`. The code will always generate the correct output.
- ◯ No line of code at `Line 1` can ensure that this code will always generate the correct ouput.

3. Match the following.

```
a) Test-and-set     1) One thread is locked out permanently
b) Starvation       2) Access control to shared resources using counters
c) Semaphore        3) Check for a value and set it in an atomic step
d) Monitors         4) Attaches synchronization control to data
```

Choose the option that gives the most appropriate matching.

- ◯ a-1, b-2, c-3, d-4
- √ a-3, b-1, c-2, d-4
- ◯ a-3, b-2, c-4, d-1
- ◯ a-3, b-4, c-2, d-1

Bank account A has ₹1000 as balance, and bank account B has ₹500 as balance. The Java program given below transfers a sum of ₹200 from account A to account B. Based on this code, answer questions 4 and 5.

```
class Account implements Runnable{
    private int balance;
    final int TRANSFER_AMT = 200;
    public int getBalance() {
        return balance;
    }
    public void setBalance(int amt) {
        balance = amt;
    }
    public int getAmountTransfer() {
        return TRANSFER_AMT;
    }
    public void run() {
    }
}
class SourceAccount extends Account{
    public SourceAccount(int amt) {
        setBalance(amt);
    }
    public void run() {
        setBalance(getBalance()-getAmountTransfer());
    }
}
class TargetAccount extends Account{
    public TargetAccount(int amt) {
        setBalance(amt);
    }
    public void run() {
        setBalance(getBalance()+getAmountTransfer());
    }
}
public class BankTransfer {
    public void transfer(SourceAccount src, TargetAccount tgt) {
        Thread t1 = new Thread(src);
        Thread t2 = new Thread(tgt);
        t1.start();
        t2.start();
    }
    public static void main(String[] args) {
        Account A = new SourceAccount(1000);
```

```
41          Account B = new TargetAccount(500);
42          BankTransfer bt = new BankTransfer();
43          bt.transfer((SourceAccount)A, (TargetAccount)B);
44          System.out.println(A.getBalance()+" "+B.getBalance());
45      }
46  }
```

4. Assume that the program terminates normally. Immediately after the execution of which of the following line/s of code, are we guaranteed to see the sum of balances in account A and account B to be equal to ₹1500?

   - ◯ Line 36
   - ◯ Line 37
   - ◯ Line 44
   - √ None of the above

5. From among the given options, choose the correct options for the method `run()` of both `SourceAccount` and `TargetAccount` such that at the end of Line 44, we see that the sum of the balances of account A and account B is ₹1500. Assume that P(S) and V(S) are atomic functions that acquire and release, respectively, the control of execution.

√
```
//SourceAccount
public void run() {
    P(S);
    setBalance(getBalance()
        -getAmountTransfer());
    V(S);
}
```

```
//TargetAccount
public void run() {
    P(S);
    setBalance(getBalance()
            +getAmountTransfer());
    V(S);
}
```

◯
```
//SourceAccount
public void run() {
    int b = getBalance();
    int a = getAmountTransfer();
    P(S);
    setBalance(b - a);
    V(S);
}
```

```
//TargetAccount
public void run() {
    int b = getBalance();
    int a = getAmountTransfer();
    P(S);
    setBalance(b  + a);
    V(S);
}
```

```
//SourceAccount
public void run() {
    P(S);
    int b = getBalance();
    int a = getAmountTransfer();
    setBalance(b - a);
    V(S);
}
```

```
//TargetAccount
public void run() {
    P(S);
    int b = getBalance();
    int a = getAmountTransfer();
    setBalance(b  + a);
    V(S);
}
```

√

○ P(S) and V(S) are insufficient to ensure consistency of data in the given example.

6. Since `Runnable` is a functional interface, the method `run()` can be implemented as a lambda function. Identify the appropriate option(s) to fill in the blank at `LINE 1`, which enable(s) the thread to print the words of the given string `text` on separate lines.

```
import java.util.*;
import java.util.stream.*;
public class FClass{
    public static void main(String[] args){
        var text = "solutions based on test-and-set are low level and
                            prone to programming errors";
        List<String> words = List.of(text.split(" "));
        Thread t = new Thread(__________________);    //LINE 1
        t.start();
    }
}
```

√ `() -> words.stream().forEach(System.out::println)`

◯ `(words) -> words.stream()`
`.forEach(System.out::println)`

◯ `words -> { for(int i = 0; i < words.length; i++)`
`System.out.println(words.get(i));`
`}`

√ `() -> { for(String s : words)`
`System.out.println(s);`
`}`

Solution: Since the `run()` method takes no argument as input, option-2 and -3 are wrong.

7. Consider the code given below.

```
class ProdList{
    private String[] items = {"pen", "pencil", "paper"};
    public void show(){
        for(String s : items)
            System.out.print(s + " ");
    }
}
class PrlProdList ____________________________ {    //LINE 1
    public void run(){
        show();
    }
}
public class FClass{
    public static void main(String[] args){
        Thread t1 = new Thread(new PrlProdList());
        Thread t2 = new Thread(new PrlProdList());
        t1.start();
        t2.start();
    }
}
```

Identify the appropriate option to fill in the blank at `LINE 1`, such that the threads `t1` and `t2` execute the `show()` function in parallel.

- ◯ `implements Runnable`
- ◯ `extends ProdList`
- ◯ `extends ProdList, Thread`
- √ `extends ProdList implements Runnable`

Solution: Since, `PrlProdList` uses the `show()` method in `ProdList` without any object reference, it must inherit `ProdList` class. `PrlProdList` class object is passed as argument in `Thread` connstructor, which must be an `Runnable` object. Thus, `PrlProdList` must implements `Runnable`.

8. Consider the code given below.

```
class PrlCls1 extends Thread{
    public void run(){
        for(int i = 1; i <= 10; i++){
            System.out.print(i + " ");
            try{
                Thread.sleep(1000);
            }
            catch(InterruptedException e){}
        }
    }
}
class PrlCls2 extends Thread{
    public void run(){
        for(int i = 11; i <= 20; i++){
            System.out.print(i + " ");
            try{
                Thread.sleep(1000);
            }
            catch(InterruptedException e){}
        }
    }
}
public class Main{
    public static void main(String[] args){
        Thread t1 = new PrlCls1();
        Thread t2 = new PrlCls2();
        t1.run();
        t2.start();
    }
}
```

Choose the correct option regarding the code.

- √ It always prints 1 to 10 first, followed by 11 to 20.
- ◯ It always prints 11 to 20 first, followed by 1 to 10.
- ◯ It may prints 11 to 20 and 1 to 10 in an interleaved manner.
- ◯ It generates a compiler error because the method `run()` cannot be invoked explicitly.

Solution: Since, the `main()` executes the `run()` method first and then starts the thread `t`, it always prints 1 to 10 first, followed by 11 to 20.

9. Suppose there are two threads T1 and T2. T1 will execute a code to deposit ₹100 to an account and thread T2 will execute a code to withdraw ₹200 from the same account. Two semaphores `A` and `B` are used for attaining mutual exclusion in this process. It is given that initially the values of both `A` and `B` are 1 and the initial balance of the account is ₹500.
Consider the following code executed by T1 and T2 threads concurrently.
T1:

```
Line No.   Instructions
1.           P(A)
2.           temp1 = balance
3.           temp1 = temp1 + 100
4.           P(B)
5.           balance = temp1
6.           V(B)
7.           V(A)
```

T2:

```
Line No.   Instructions
1.           P(B)
2.           temp2 = balance
3.           temp2 = temp2 - 200
4.           P(A)
5.           balance = temp2
6.           V(A)
7.           V(B)
```

Choose the correct option regarding the program execution.

- ◯ After execution is finished the balance can be either ₹600 or ₹400.
- ◯ After execution is finished the balance can be either ₹300 or ₹600.
- √ After execution is finished the balance will be ₹400.
- √ The program attains mutual exclusion but may run into a deadlock.

Solution:
In this example there can be three possibilities.

1. T1 starts and completes its execution and then T2 starts
2. T2 starts and completes its execution and then T1 starts

3. The program will run into a deadlock, if T1 executes Line 1 and then T2 executes Line 1.

In the first two cases balance would be Rs 400 after program completes. Therefore the account will never have a balance of Rs 600 or Rs 300 i.e. in other words if deadlock does not occur then it can be said, none of the updates are lost. One should be careful while using semaphores

10. Consider the following code.

```
class ATM implements Runnable{
  private String name;
  static double balance = 100.0;
  public ATM(String name){
    this.name = name;
  }
  public void deposit(double a){
    balance = balance + a;
  }
  public void withdraw(double a){
    balance = balance - a;
  }
  public void displayBalance(){
    System.out.println(balance);
  }
  public void run(){
    if(name.equals("deposit")){
        deposit(100);
        displayBalance();
    }
    if(name.equals("withdraw")){
        withdraw(200);
        displayBalance();
    }
  }
}
public class cardUser {
  public static void main(String[] args) {
    ATM m1 = new ATM("deposit");
    ATM m2 = new ATM("withdraw");
    Thread t1 = new Thread(m1);
    Thread t2 = new Thread(m2);
    t1.start();
    t2.start();

  }
}
```

Select the most appropriate option.

- ◯ The program may generate the output:
  -100.0
  0.0

- ◯ The program may generate the output:
  0.0
  0.0
- ◯ The program may generate the output:
  200.0
  0.0
- √ All the above

BSCCS2005: Graded with Solutions
Week 11

1. Consider the code given below.

```
public class PrlSequence extends Thread{
    int init;
    public PrlSequence(int i) {
        init = i;
    }
    public void run(){
        for(int i = init; i <= init + 5; i++) {
            System.out.print(i + " ");
            try {
                sleep(500);
            }
            catch(InterruptedException e) {}
        }
    }
}

public class FClass{
    public static void main(String[] args) throws InterruptedException{
        Thread th1 = new  PrlSequence(10);
        Thread th2 = new  PrlSequence(20);
        Thread th3 = new  PrlSequence(30);
        th1.start();
        th2.start();
        th1.join();
        th2.join();
        th3.start();
    }
}
```

Choose the correct option regarding the code.

- ◯ It may print 10 to 15, 20 to 25, and 30 to 35 in an interleaved manner.
- √ It may print 10 to 15 and 20 to 25, in an interleaved manner, followed by printing 30 to 35.
- ◯ It prints 10 to 15 first, followed by 20 to 25, and followed by printing 30 to 35.
- ◯ It prints 10 to 15 first, followed by 20 to 25, and 30 to 35 in an interleaved manner.

Solution: The `th1.join()` method ensures that the main thread wait for `th1` to complete,

The `th2.join()` method ensures that the main thread wait for `th2` to complete, then starts `th3`.
Thus, it prints 10 to 15 and 20 to 25 in interleaved fashion, and then prints 30 to 35.

2. Consider the following code which tries to simulate a producer-consumer relationship on cakes.

```
class CakeOperation {
    int available_cake_count=0;
    boolean eating_permit = false;

    synchronized void eatCake() {
        while(eating_permit == false) {
            try {
             wait();
            }catch(InterruptedException e) {}
        }
        available_cake_count--;
        System.out.println("Ate one cake. Cakes left: "+ available_cake_count);

        // SEGMENT 1
    }
    synchronized void makeCake() {

        // SEGMENT 2

        available_cake_count++;
        System.out.println("Made one cake. Cakes left: "+ available_cake_count);
        eating_permit = true;
        notify();
    }
}
class Baker implements Runnable {
    CakeOperation obj;
    Baker(CakeOperation o) {
        obj = o;
        Thread Producer = new Thread(this);
        Producer.start();
    }
    public void run() {
        int cake_number = 1;
        while(cake_number <= 2) {
            obj.makeCake();
            cake_number++;
        }
    }
}
class Consumer implements Runnable {
```

```
    CakeOperation obj;
    Consumer(CakeOperation o) {
        obj = o;
        Thread consumer = new Thread(this);
        consumer.start();
    }
    public void run() {
        int iteration = 0;
        while(iteration < 2) {
            obj.eatCake();
            iteration++;
        }
    }
}
public class Test2 {
    public static void main(String args[]) {
        CakeOperation obj  = new CakeOperation();
        Baker b1 = new Baker(obj);
        Consumer c1 = new Consumer(obj);
    }
}
```

If the given code always generates the following output then what should be written in place of `SEGMENT 1` and `SEGMENT 2` ?

Output

```
Made one cake. Cases left: 1
Ate one cake. Cases left: 0
Made one cake. Cases left: 1
Ate one cake. Cases left: 0
```

SEGMENT 1:

√
```
eating_permit = false;
notify();
```

SEGMENT 2:

```
if(eating_permit == true) {
    try {
        wait();
    }catch(InterruptedException e) {}
}
```

○ *SEGMENT 1:*

```
eating_permit = true;
notify();
```

SEGMENT 2:

```
if(eating_permit == true) {
    try {
        wait();
    }catch(InterruptedException e) {}
}
```

○ *SEGMENT 1:*

```
eating_permit = false;
notify();
```

SEGMENT 2:

```
if(eating_permit == false) {
    try {
        wait();
    }catch(InterruptedException e) {}
}
```

○ *SEGMENT 1* and *SEGMENT 2* can be left blank, the code will produce the required output because the methods are synchronized.

Solution: Detailed sol

3. Consider the following code which converts temperatures given in Centigrade scale to Fahrenheit scale and vice versa, using two threads. Once the conversion is finished, the program should check whether there is a temperature value in the array which is the same in both the scales.

```
import java.util.*;
class Centigrade extends Thread{
    double[] f_temp = null;
    public Centigrade(double[] arr) {
        f_temp = arr;
    }
    public void run() {
        // converts the temperatures in f_temp to centigrade
        for(int i=0;i<f_temp.length;i++) {
            f_temp[i] = ((f_temp[i] - 32)/9.0)*5;
        }
    }
}
class Fahrenheit extends Thread{
    double[] c_temp = null;
    public Fahrenheit(double[] arr) {
        c_temp = arr;
    }
    public void run() {
        // converts the temperatures in c_temp to fahrenheit
        for(int i=0;i<c_temp.length;i++) {
            c_temp[i] = ((9.0/5.0)*c_temp[i])+32.0;
        }
    }
}
public class Test{
    public static void main(String args[]) {
        double temp_in_fahrenheit[] = {29.67, -40.0, 50.0, 32.0};
        double temp_in_centigrade[] = {100.0, -92.54, 32.0, -40.0};
        Centigrade cobj = new Centigrade(temp_in_fahrenheit);
        Fahrenheit fobj = new Fahrenheit(temp_in_centigrade);
        cobj.start();
        fobj.start();

        //***** SEGMENT 1 ******

        // Checking if a temperature is same in both centigrade and fahrenheit
        for(int i=0;i<4;i++) {
            for(int j=0;j<4;j++) {
```

```
                if(temp_in_fahrenheit[i] == temp_in_centigrade[j])
                    System.out.println(temp_in_centigrade[j]);
            }
        }
    }
}
```

- 40 is the only temperature which is same in both the scales. Choose all the correct options which should be written in place of SEGMENT 1, so that this program produces the result as -40 on every execution.

√
```
try {
    cobj.join();
    fobj.join();
}catch(InterruptedException e) {}
```

◯
```
try {
    Thread.currentThread().wait();
}catch(InterruptedException e) {}
```

◯
```
try {
    Thread.sleep(2);
}catch(InterruptedException e) {}
```

◯ `SEGMENT 1` can be left blank, the code will anyway produce -40 on every execution.

Solution: Detailed sol

4. Consider the code given below.

```
class TicketMyshow implements Runnable{
    int available = 10;
    int wanted;
    public TicketMyshow(int w) {
        wanted = w;
    }
    public synchronized void run() {
        String name = Thread.currentThread().getName();
        System.out.println("AvailableTickets : " + available + " Wanted : "+wanted);
        if (wanted <= available){
            System.out.println(name+" booked " + wanted + " tickets");
            available = available - wanted;
        }
        else{
            System.out.println("Sorry no tickets for " + name);
        }
    }
}
public class BookTicket{
    public static void main(String[] args){
        TicketMyshow ticket = new TicketMyshow(5);
        Thread thread1 = new Thread(ticket,"Jock");
        thread1.start();
        Thread thread2 = new Thread(ticket, "John");
        thread2.start();
        Thread thread3 = new Thread(ticket, "Virat");
        thread3.start();
    }
}
```

Which of the following options is/are possible result/s of the above code?

- ◯ `AvailableTickets : 10 Wanted : 5`
  `Virat booked 5 tickets`
  `AvailableTickets : 5 Wanted : 5`
  `Jock booked 5 tickets`
  `AvailableTickets : 0 Wanted : 5`
  `Sorry no tickets for John`
- ◯ `AvailableTickets : 10 Wanted : 5`
  `Jock booked 5 tickets`
  `AvailableTickets : 5 Wanted : 5`
  `John booked 5 tickets`

```
AvailableTickets : 0 Wanted : 5
Sorry no tickets for Virat
```

○
```
AvailableTickets : 10 Wanted : 5
John booked 5 tickets
AvailableTickets : 5 Wanted : 5
Virat booked 5 tickets
AvailableTickets : 0 Wanted : 5
Sorry no tickets for Jock
```

√ All of the above

Solution: We cannot predict order of thread's execution, but all threads are synchronized, among three threads one thread will not get any ticket.

5. Consider the code given below.

```
class BookTicket{
    int available=3;
    synchronized void extract(String name,int book){
        if (book<=available){
            System.out.println(name + " booked " + book + " ticket.");
            available = available - book;
        }
        else {
            System.out.println(name + ", you can not book " + book + " ticket.");
            System.out.println("Tickets available: " + available);
        }
    }
}
class Passenger implements Runnable{
    BookTicket bt;
    String name;
    int number;
    Passenger(BookTicket b, String n, int num){
        this.bt = b;
        this.name = n;
        this.number = num;
    }
    public void run(){
        bt.extract(name, number);
    }
}
public class Test {
    public static void main(String[] args){
        BookTicket obj = new BookTicket();
        BookTicket obj1 = new BookTicket();
        BookTicket obj2 = new BookTicket();
        Thread t1 = new Thread(new Passenger(obj, "Sun", 2));
        Thread t2 = new Thread(new Passenger(obj1, "Moon", 1));
        Thread t3 = new Thread(new Passenger(obj2, "Earth", 2));
        t1.start();
        t2.start();
        t3.start();
    }
}
```

What will be the possible output(s)?

√ `Moon booked 1 ticket.`
`Sun booked 2 ticket.`
`Earth booked 2 ticket.`

◯ `Sun booked 2 ticket.`
`Moon booked 1 ticket.`
`Earth, you can not book 2 ticket.`
`Tickets available: 0`

◯ This program generates `InterruptedException` at runtime.

◯ This program generates `IllegalMonitorStateException` at runtime.

Solution: `Synchronization` provides an object level lock and allows only 1 thread at a time to invade the critical region. An object can have multiple threads but the sensitive area can only be accessed by 1 thread at a time.

In this program multiple threads are associated with multiple `Objects`. So at a time, 1 thread from each object will get an opportunity to invade the critical region and create race condition. It is highly unlikely that this program will generate correct output fight shying the race condition, but this can not be guaranteed. Hence to prevent race condition class level lock is required.

6. Consider the code given below.

```
import java.util.concurrent.locks.*;
class DemoLock{
    ReentrantLock lck = new ReentrantLock();
    public void display(String name){
        lck.lock();
        try{
            for(int i = 1; i < 4; i++){
            System.out.print(i + ":" + name + " ");
            }
            System.out.print("\n");
        }
        finally{
            lck.unlock();
        }
    }
}
class Example extends Thread{
    DemoLock l_obj;
    String str;
    Example(DemoLock o, String str){
        this.l_obj = o;
        this.str = str;
    }
    public void run(){
        l_obj.display(str);
    }
}
public class FClass{
    public static void main(String[] args){
        DemoLock obj = new DemoLock();
        Example e = new Example(obj, "Sun");
        Example e2 = new Example(obj, "Moon");
        Example e3 = new Example(obj, "Earth");
        e.start();
        e2.start();
        e3.start();
    }
}
```

What is/are the possible output/s?

√ 1:Sun 2:Sun 3:Sun
1:Earth 2:Earth 3:Earth
1:Moon 2:Moon 3:Moon

◯ 1:Moon 2:Moon 1:Earth 1:Sun 2:Earth 3:Moon
3:Earth
2:Sun 3:Sun

◯ 1:Earth 1:Sun 1:Moon
2:Moon 2:Sun 2:Earth
3:Sun 3:Moon 3:Earth

√ 1:Sun 2:Sun 3:Sun
1:Moon 2:Moon 3:Moon
1:Earth 2:Earth 3:Earth

7. Consider the code given below.

```
import java.util.logging.*;
class Example extends Thread{
    Thread t;
    Example(Thread t){
        this.t = t;
    }
    public void run(){
        try {
            t.join();
        } catch (InterruptedException ex) {
            Logger.getGlobal().log(Level.SEVERE, ex.getMessage());
        }
        for(int i = 1; i < 3; i++){
            System.out.print(i + " ");
        }
    }
}
public class FClass{
    public static void main(String[] args) throws InterruptedException{
        Thread t1 = Thread.currentThread();
        Example t2 = new Example(t1);
        t2.start();
        for(int i = 3; i <= 5; i++){
            System.out.print(i + " ");
        }
    }
}
```

What is a possible output?

- ◯ 1 2 3 4 5
- √ 3 4 5 1 2
- ◯ 3 4 5
- ◯ All of them

8. Consider the code given below.

```
import java.util.*;
class PrlTest extends Thread{
    Map<String, Integer> mp;
    Thread th;
    PrlTest(Map<String, Integer> ic,Thread t){
        this.mp = ic;
        this.th = t;
    }
    public void run(){
        try {
            th.join();
        } catch (InterruptedException ex) {}
        mp.put("D", 4);
    }
}
public class FClass{
    public static void main (String[] args) throws InterruptedException{
        Thread t1 = Thread.currentThread();
        Map<String, Integer> icMap = new LinkedHashMap<String, Integer>();
        String[] str = {"A", "B", "C"};
        Integer[] arr = {1, 2, 3};
        for(int i = 0; i < str.length; i++){
           icMap.put(str[i], arr[i]);
        }
        PrlTest t2 = new PrlTest(icMap, t1);
        t2.start();
        t2.join();
        for(Map.Entry m : icMap.entrySet()){
           System.out.println(m.getKey() + " => "+ m.getValue());
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ This program prints
  ```
  A => 1
  B => 2
  C => 3
  ```
- √ This program results in a deadlock.
- ◯ This program prints

```
A => 1
B => 2
C => 3
D => 4
```

◯ This program prints all 4 elements: `A => 1, B => 2, C => 3` and `D => 4`. However, the order of the elements is undetermined.

9. Consider the code given below.

```
import java.util.*;
import java.util.concurrent.*;
class Example extends Thread{
    Map siMap;
    Example(Map m){
        this.siMap = m;
    }
    public void run(){
        siMap.put("D",4);
    }
}
public class FClass{
    public static void main (String[] args) {
        Map<String, Integer> siMap = new ConcurrentHashMap();
        String[] str = {"A", "B", "C"};
        Integer[] arr = {1, 2, 3};
        for(int i = 0; i < str.length; i++){
                siMap.put(str[i],arr[i]);
        }
        Example t = new Example(siMap);
        t.start();
        Set s = siMap.entrySet();
        Iterator itr = s.iterator();
        while(itr.hasNext()){
            Map.Entry m = (Map.Entry)itr.next();
            System.out.println(m.getKey() + " => " + m.getValue());
        }
    }
}
```

Which of the following is NOT true about the given code.

√ This program may generate `ConcurrentModificationException`.

◯
```
A => 1
B => 2
C => 3
D => 4
```

◯
```
D => 4
A => 1
B => 2
C => 3
```

◯ `A => 1`
`B => 2`
`C => 3`

BSCCS2005: Graded with Solutions
Week 12

1. Consider the code given below.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class ButtonPanel extends JPanel{
    private JButton redButton;
    public ButtonPanel(){
        redButton = new JButton("Red");
        redButton.addActionListener(
            //CODE SEGMENT
        );
        add(redButton);
    }
}
class ButtonFrame extends JFrame implements WindowListener{
    private Container contentPane;
    public ButtonFrame(){
        setTitle("Button Demo");
        setSize(300, 200);
        addWindowListener(this);
        contentPane = this.getContentPane();
        contentPane.add(new ButtonPanel());
    }
    // define seven methods for implementing WindowListener
}
public class FClass{
    public static void main(String[] args) {
        EventQueue.invokeLater(
                () -> {
                    JFrame frame = new ButtonFrame();
                    frame.setVisible(true);
                }
                );
    }
}
```

`ActionListener` is a functional interface that has the abstract method `actionPerformed()`. Since `addActionListener` requires an instance of `ActionListener` as parameter, we can supply a lambda expression. Identify the appropriate option(s) to be written in place of `CODE-SEGMENT` such that the listener sets the panel background to red when the button is clicked.

√ `(ActionEvent evt) -> {`

```
            Color c = Color.red;
            setBackground(c);
            repaint();
        }
```

◯ `() -> {`

```
            Color c = Color.red;
            setBackground(c);
            repaint();
        }
```

√ `ea -> {`

```
            setBackground(Color.red);
            repaint();
        }
```

◯ `() -> {`

```
            this.setBackground(Color.red);
            repaint();
        }
```

Solution: The abstract method `actionPerformed()` accepts an argument of type `ActionEvent`. The lambda method without any arguments is not valid.

2. Consider the code given below.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class ButtonFrame extends JFrame implements ActionListener, WindowListener{
    private Container contentPane;
    private JPanel panel;
    private JButton colBtn;
    private int i = 0;
    private String[] colors = new String[]{"red", "yellow", "green"};
    public ButtonFrame(){
        //set the JFrame of a given size
        panel = new JPanel();
        colBtn = new JButton("Color");
        colBtn.setActionCommand(colors[i]);
        //add colBtn to panel, panel to the current contentPane
        colBtn.addActionListener(this);
        addWindowListener(this);
    }
    public void actionPerformed(ActionEvent evt){
        Color c;
        String s = evt.getActionCommand();
        if(s == "red")
            c = Color.red;
        else if(s == "yellow")
            c = Color.yellow;
        else
            c = Color.green;
        panel.setBackground(c);
        panel.repaint();
        i = (i + 1) % 3;
        colBtn.setActionCommand(colors[i]);
    }
    // define seven methods for implementing WindowListener
}
public class FClass{
    public static void main(String[] args) {
        EventQueue.invokeLater(
                () -> {
                    JFrame frame = new ButtonFrame();
                    frame.setVisible(true);
                }
```

```
        );
    }
}
```

Choose the correct option regarding the code.

- ◯ The initial color of the panel is red and it remains unchanged with the button clicks.
- ◯ The initial color of the panel is red and it toggles between red and yellow with the button clicks.
- √ The initial color of the panel is red, with next button click it becomes yellow, with one more button click it becomes green, and it goes on repeating.
- ◯ The initial color of the panel is red, with next button click it becomes yellow, with one more button click it becomes green, and followed by any arbitrary color.

Solution: The color of the panel is decided by the strings in `String[] colors = new String[]"red", "yellow", "green";`.
Initially the color is `colors[0]` which is red.
For each button click `i` becomes `i = (i + 1) % 3;`, i.e. $1, 2, 0, 1, 2, 0, 1, 2, \cdots$. Thus, the colors becomes "red", "yellow", "green", repeatedly.

3. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class GUITest extends JFrame implements ActionListener{
    JButton b1,b2,b3;
    JPanel panel;
    public GUITest(){
        panel=new JPanel();
        b1=new JButton("Red");
        b2=new JButton("Green");
        b3=new JButton("Blue");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        panel.add(b1);
        panel.add(b2);
        panel.add(b3);
        add(panel,"South");
        setVisible(true);
        setSize(400,400);
    }
    public void actionPerformed(ActionEvent e) {
        ----------------------
        *****CODE SEGMENT*****
        ----------------------
    }
    public static void main(String[] args){
        new GUITest();
    }
}
```

Choose the correct code segment inside method `actionPerformed()` such that whenever either of the three buttons (Red/Green/Blue) is clicked, the panel background color changes accordingly.

○
```
if(e.equals(b1))
    panel.setBackground(Color.red);
if(e.equals(b2))
    panel.setBackground(Color.green);
if(e.equals(b3))
    panel.setBackground(Color.blue);
```

√ `if(e.getSource().equals(b1))`

```
    panel.setBackground(Color.red);
if(e.getSource().equals(b2))
    panel.setBackground(Color.green);
if(e.getSource().equals(b3))
    panel.setBackground(Color.blue);
```

◯
```
if(e.getSource().equals("Red"))
    panel.setBackground(Color.red);
if(e.getSource().equals("Green"))
    panel.setBackground(Color.green);
if(e.getSource().equals("Blue"))
    panel.setBackground(Color.blue);
```

◯
```
if(e.equals("Red"))
    panel.setBackground(Color.red);
if(e.equals("Green"))
    panel.setBackground(Color.green);
if(e.equals("Blue"))
    panel.setBackground(Color.blue);
```

Solution:

4. Consider the code given below.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class ColorThread extends Thread{
    JPanel inputPanel;
    Color inputCol;
    Thread wTh;
    public ColorThread(JPanel ip, Color col, Thread th) {
        inputPanel = ip;
        inputCol = col;
        wTh = th;
    }
    public void run() {
        try {
            if(wTh != null)
                wTh.join();
            inputPanel.setBackground(inputCol);
            sleep(1000);
        }catch(InterruptedException e) {}
    }
}
public class FClass implements ActionListener{
    JFrame frm;
    JPanel inputPanel;
    JButton btnStart;
    FClass(){
        frm = new JFrame("Traffic Light");
        frm.setSize(200, 200);
        btnStart = new JButton("Start");
        btnStart.addActionListener(this);
        inputPanel = new JPanel();
        inputPanel.add(btnStart);
        frm.add(inputPanel);
        frm.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        Thread th1 = new ColorThread(inputPanel, Color.red, null);
        Thread th2 = new ColorThread(inputPanel, Color.yellow, th1);
        Thread th3 = new ColorThread(inputPanel, Color.green, th2);
        th1.start();
        th2.start();
```

```
        th3.start();
    }
    public static void main(String[] args){
        new FClass();
    }
}
```

Choose the option that correctly describes what happens if the button labelled "Start" is clicked.

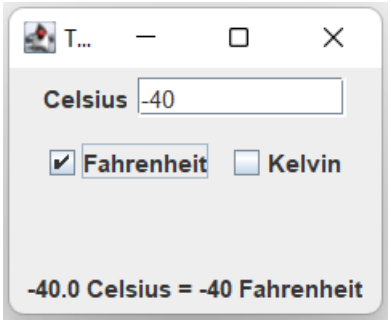
- √ The background color of panel `inputPanel` becomes red first, followed by yellow, further followed by green.
- ◯ The background color of panel `inputPanel` becomes red first, followed by yellow, further followed by green. The same sequence is repeated until the program terminates.
- ◯ The background color of panel `inputPanel` becomes red, yellow and green. However, the sequence in which the colors get rendered cannot be predicted.
- ◯ The background of panel `inputPanel` takes only one color. It becomes either red or yellow or green.

5. Consider the code given below.

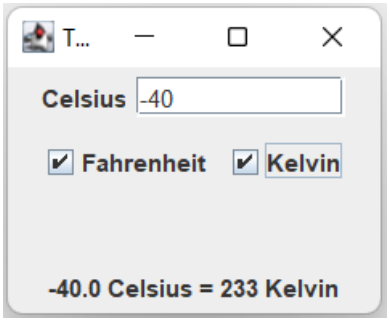
```
import javax.swing.*;
import java.awt.event.*;
public class FClass implements ActionListener{
    JFrame frm;
    JLabel lblCel;
    JTextField txtCel;
    JCheckBox chbFahr, chbKelv;
    JLabel lblMsg;
    FClass(){
        frm = new JFrame("Temperature conversion");
        frm.setSize(300, 200);
        lblCel = new JLabel("Celsius");
        txtCel = new JTextField(10);
        chbFahr = new JCheckBox("Fahrenheit");
        chbKelv = new JCheckBox("Kelvin");
        chbFahr.addActionListener(this);
        chbKelv.addActionListener(this);
        lblMsg = new JLabel();

        //add inputPanel, outputPanel and btnPanel to
        //the "North", "Bottom" and "Center" of the JFrame
        //add lblCel and txtCel to inputPanel
        //add lblMsg to outputPanel
        //add chbFahr and chbKelv to btnPanel
        frm.setVisible(true);
    }
    public static void main(String[] args){
        new FClass();
    }
    public void actionPerformed(ActionEvent e){
        if(chbFahr.isSelected()){
            double valCels = Double.parseDouble(txtCel.getText());
            int valFahr = (int)(((valCels * 9) / 5) + 32);
            lblMsg.setText(valCels + " Celsius = "+ valFahr + " Fahrenheit");
        }
        if(chbKelv.isSelected()){
            double valCels = Double.parseDouble(txtCel.getText());
            int valKelv = (int)(valCels + 273);
            lblMsg.setText(valCels + " Celsius = "+ valKelv + " Kelvin");
        }
        if(chbFahr.isSelected() && chbKelv.isSelected()){
            lblMsg.setText("Invalid option");
```

```
        }
    }
}
```

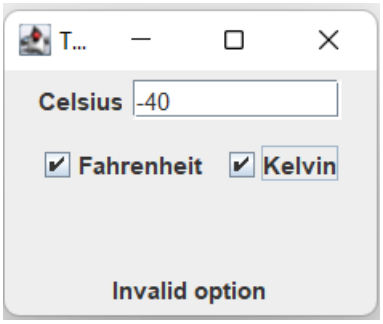
Which of the possible GUI(s) is/are generated by the given code.



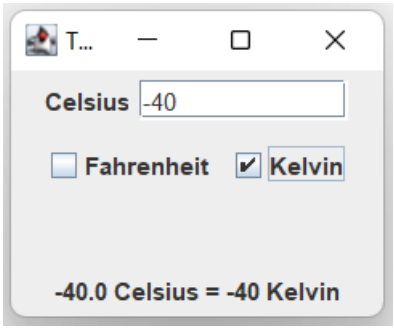
√



○



√

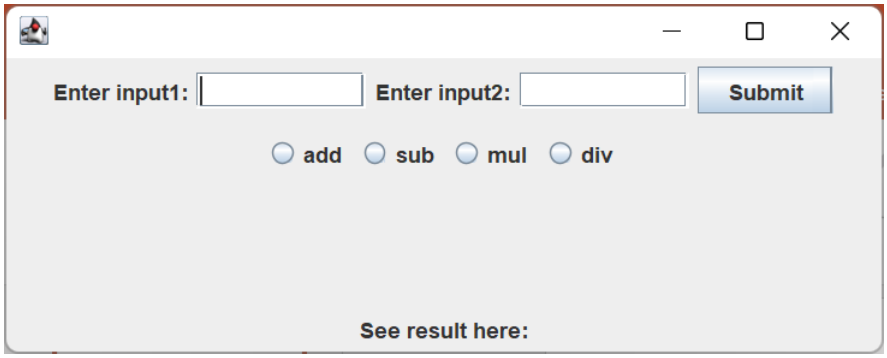


○

6. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
public class Calculator extends JFrame{
    JPanel inputPanel,outputPanel,opPanel;
    JLabel label1,label2,label3;
    JRadioButton add,sub,mul,div;
    JTextField input1,input2;
    JButton button;
    public Calculator() {
        label1=new JLabel("Enter input1:");
        label2=new JLabel("Enter input2:");
        input1=new JTextField(10);
        input2=new JTextField(10);
        button=new JButton("Submit");
        inputPanel=new JPanel();
        inputPanel.add(label1);
        inputPanel.add(input1);
        inputPanel.add(label2);
        inputPanel.add(input2);
        inputPanel.add(button);
        //LINE 1
        add=new JRadioButton("add");
        sub=new JRadioButton("sub");
        mul=new JRadioButton("mul");
        div=new JRadioButton("div");
        opPanel=new JPanel();
        opPanel.add(add);
        opPanel.add(sub);
        opPanel.add(mul);
        opPanel.add(div);
        //LINE 2
        label3=new JLabel("See result here:");
        outputPanel=new JPanel();
        outputPanel.add(label3);
        //LINE 3
        setVisible(true);
        setSize(500,200);
    }
    public static void main(String[] args) {
        new Calculator();
    }
}
```

Choose the correct options for `LINE` 1, 2 and 3, such that the above program produces the GUI given below:



- ◯ `LINE 1:add(inputPanel,"Center");`
  `LINE 2:add(opPanel,"North");`
  `LINE 3:add(outputPanel,"South");`
- ◯ `LINE 1:add(inputPanel,"North");`
  `LINE 2:add(opPanel,"South");`
  `LINE 3:add(outputPanel,"Center");`
- √ `LINE 1:add(inputPanel,"North");`
  `LINE 2:add(opPanel,"Center");`
  `LINE 3:add(outputPanel,"South");`
- ◯ `LINE 1:add(inputPanel,"South");`
  `LINE 2:add(opPanel,"Center");`
  `LINE 3:add(outputPanel,"North");`

Solution: To obtain the GUI given, we should add `inputPanel` to the `South`, `opPanel` to the `Center` and `outputPanel` to the `North` of the frame.

BSCCS2005: Practice Assignment with Solutions
Week 1

1. Which of the following statements is/are true about the *type of a variable*?

[MSQ:2points]

- ◯ In Java, the type of a variable depends on the value assigned to it.
- √ In Python, the type of a variable depends on the value assigned to it.
- ◯ In Python, the type of a variable can be determined by its data type at compile time itself.
- √ In Java, the type of a variable can be determined by its data type at compile time itself.

2. Which of the following statements is/are true about *dynamic lookup*?

[MCQ : 2 points]

- ◯ Dynamic lookup means different objects respond in the same way to the same message.
- √ Dynamic lookup means different objects respond in different ways to the same message.
- ◯ Dynamic lookup means the same object responds in different ways to the same message.
- ◯ Dynamic look up hides internal details of an object.

Solution: Dynamic lookup is a feature of object-Oriented programming languages. It says that different objects respond to a same message in different ways.

3. Which of the following features of an object-Oriented programming language hides implementation details?

[MCQ : 2 points]

- ◯ Dynamic lookup
- √ Abstraction
- ◯ Subtyping
- ◯ Inheritance

Solution: Abstraction is a feature of object-oriented programming languages that hides implementation details.

4. Suppose ‘a’ and ‘b’ are two objects such that object ‘a’ has all the functionalities of object ‘b’ along with some extra functionalities. Then, which of the following statements is/are true?

[MCQ : 2 points]

- ◯ We can make object ‘a’ as a subtype of object ‘b’, and use object ‘b’ wherever object ‘a’ is required.
- ◯ We can make object ‘b’ as a subtype of object ‘a’, and use object ‘b’ wherever object ‘a’ is required.
- √ We can make object ‘a’ as a subtype of object ‘b’ and use object ‘a’ wherever object ‘b’ is required.
- ◯ We can make object ‘b’ as a subtype of object ‘a’, and use object ‘a’ wherever object ‘b’ is required.

5. Which among the following statements is/are true about writing programs in low level languages and high level languages?

[MSQ: 2points]

- √ Writing a program in a low level language is error-prone.
- ◯ A program written in a low level language is more readable and easy to maintain.
- ◯ A programmer has more control over how a code is mapped to the machine architecture when the code is written in a high level language.
- √ Language translators like compilers and interpreters are required to convert a program in a high level language into a low level language.

Solution: The correct statements are as follows:

1. Writing a program in low level language is error-prone.
2. Considering the factors readability and maintainability of the code, high level languages are more effective.
3. Writing code in high level language is less efficient since the programmer has limited control over how the code is mapped to the machine architecture.
4. Language translators like compilers and interpreters are required to convert a program in a high level language into a low level language.

6. Which among the following statements is/are true about imperative and declarative programming style? [MSQ: 2points]

   - ◯ An imperative program focuses on what the output is and how the output is related to the input, whereas a declarative program focuses on how to obtain the output.
   - √ An imperative program describes the steps of instructions to produce the output, whereas a declarative program describes what the output should be.
   - √ An imperative program depends on a large set of instructions that manipulate the intermediate variables, whereas a declarative program may totally avoid using intermediate variables.
   - ◯ Imperative programs use an inductive structure and allows defining functions in terms of smaller functions. Thus, imperative programs are more readable as compared to declarative programs.

Solution: The correct statements are as follows:

1. A declarative program focus on "what is the output? and how it is related to input?", whereas an imperative program focus on "how to get the output?"
2. An imperative program describes the steps of instructions to produce the output, whereas a declarative program describes what computation should produce.
3. An imperative program depends on large set of instructions that manipulates the intermediate variables, whereas a declarative program may totally avoid using intermediate variables,
4. Declarative programs uses an inductive structure and allows defining functions in terms of smaller functions compare to the imperative programs. Thus, the functions in imperative programs are more readable (or understandable).

7. Which of the following statements is/are true about *variables*?

[MSQ: 2points]

- ◯ Any variable can be accessed if it is on the stack.
- √ A variable should be in the correct scope in order to be accessible.
- ◯ Statically and dynamically created data is stored on the stack.
- √ During the lifetime of a variable, there may be times when it is not in scope.

Solution:

- variable can be out of scope even if it is in stack.
- **variable must be in its scope in order to access it**.
- variables which outlives the activation record are stored in heap.
- **If the variable is stored in heap, its scope outlives the function in which it is called, but the scope is within the function in which it is defined.**

8. Let **A** be an object of type **Animal** and **B** be an object of type **Bird**.

[MSQ: 2points]

```
class Animal:
    def __init__(self, L):
        self.Legs = L
    def walk(self):
      print("Animal walks with", self.Legs, "legs")
class Bird(Animal):
    def __init__(self, L, W):
        self.Legs = L
        self.Wings = W
    def fly(self):
        print("Bird flies")
```

Which of the following options is/are NOT valid?

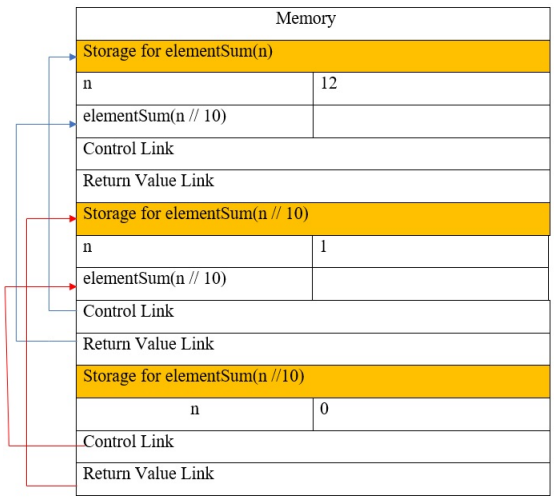
- ◯ **B**.fly( )
- √ **A**.fly( )
- ◯ **A**.walk( )
- ◯ **B**.walk( )

9. Choose the correct memory mapping for the following pseudocode:

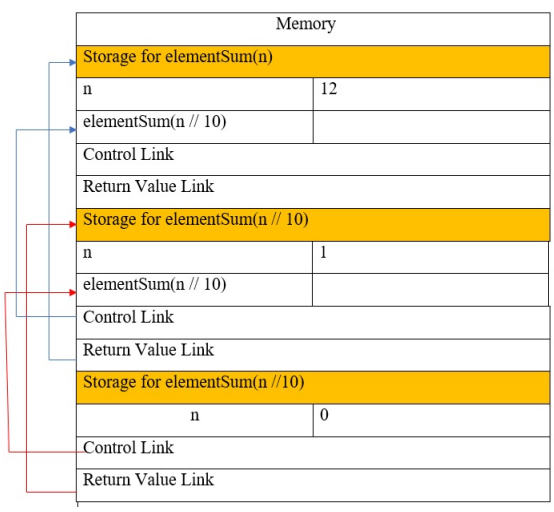
[MCQ: 2points]

```
def elementSum(n):
    if(n == 0):
        return 0
    else:
        return n % 10 + elementSum(n//10)
elementSum(12)
```

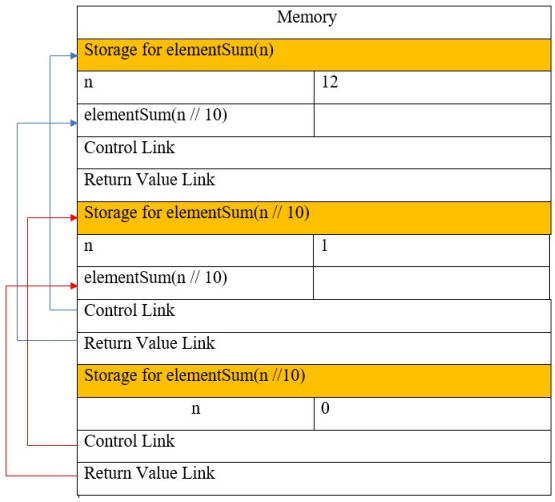
◯ Option 1



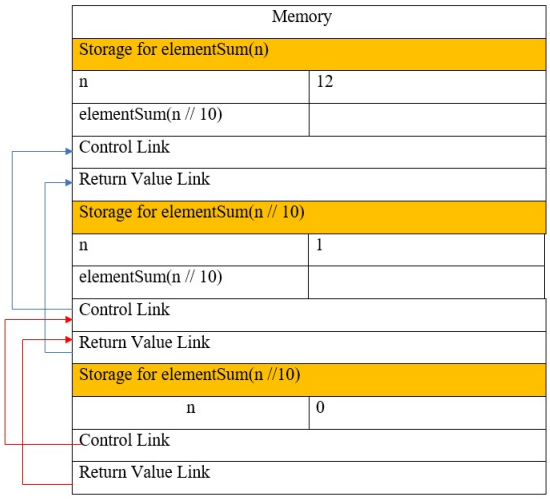
◯ Option 2



◯ Option 3



◯ Option 4



Solution: Control link points to the previous activation record. The return value link tells where to store the result.

10. Consider the following Python code.

[MCQ: 2points]

```
def area(radius):
    return(22/7*radius*radius)

def area(breadth,length):
    return(length*breadth)

print(area(7))
print(area(7,10))
```

What will be the output of the code?

- √ The code generates an error
- ◯ 154,70
- ◯ 70,70
- ◯ 154,154

Solution: This example demonstrates that Python does not support method overloading by default.

11. From among the following, choose the INCORRECT characteristic/s about Abstract Data Types.

[MSQ: 2points]

- ◯ Fixed interface
- ◯ Structured data collection
- √ Fixed implementation
- ◯ Implementation can be changed but should not affect interface

Solution: Abstract Data Types are structured collections with fixed interfaces. Changing implementation of Abstract Data Types should not affect interface

12. Let **A** be an object of **Vehicle**, **B** be an object of **Twowheeler**. What is the output of **B**.display( )? [MCQ: 2points]

```
class Vehicle:
    def display(self):
      print("Every vehicle has wheels")
class Twowheeler(Vehicle):
    def display(self):
        print("Every Two wheeler has two wheels")
```

- ◯ Error
- ◯ Every vehicle has wheels
  Every Two wheeler has two wheels
- ◯ Every vehicle has wheels
- √ Every Two wheeler has two wheels

Solution:

- **B** is subtype of **A**.
- Both **A**, **B** have display( ) method.
- **B** is calling display( ) method, so the display( ) method of **B** gets executed.

graphixs

BSCCS2005: Practice Assignment Questions with Solutions
Week {2}

{Write general instructions here}

1. What is the output of the following Java code? [MCQ:2points]

```
public class Example {
    public static void main(String[] args) {
        System.out.println("Addition of 10 and 20 is " + 10 + 20);
    }
}
```

- ◯ `Addition of 10 and 20 is 30`
- √ `Addition of 10 and 20 is 1020`
- ◯ `"Addition of 10 and 20 is " + 10 + 20`
- ◯ `Error`

Solution: string concatenation is happening here.

- "Addition of 10 and 20 is " + 10 gives "Addition of 10 and 20 is 10" here string is concatenated with an integer which results in a string.
- "Addition of 10 and 20 is 10"+ 20 gives "Addition of 10 and 20 is 1020" here string is concatenated with an integer which results in a string.

2. What is the output of the following Java code? [MCQ:2points]

```
public class FClass{
    public static void main(String[] args) {
        int[] a = {1, 2, 3};
        int[] b = a;
        System.out.print(a == b);
    }
}
```

- ◯ `a == b`
- ◯ `false`
- √ `true`
- ◯ `{1, 2, 3} == {1, 2, 3}`

Solution: It prints true. The `==` operator compares whether `a` and `b` are referring to the same memory location.

3. What is the output of the following Java code? [MCQ:2points]

```
public class FClass{
    public static void main(String[] args) {
        String[] str1, str2 = {"apples", "grapes", "bananas"};
        System.out.print(str1[1]);
    }
}
```

- ◯ `"apples"`
- ◯ `"grapes"`
- ◯ {`"apples"`}
- ◯ {`"grapes"`}
- √ **Compiler Error**

Solution: `str1` is declared but not initialized. It gives an error because we are trying to access `str1[1]` which is not initialized

4. Consider the Java program below. [MCQ:2 points]

```
class FClass{
    public static void main(String[] args) {
        int i1 = 10, i2 = 29;
        double d;
        d = i1 / i2 * i2;
        System.out.print(d + " ");
        d = i2 * i1 / i2;
        System.out.print(d + " ");
    }
}
```

What will be the output?

- ◯ `0.0 0.0`
- ◯ `10.0 10.0`
- ◯ `10.0 0.0`
- √ `0.0 10.0`

Solution: The computations for statement `d = i1 / i2 * i2;` take place as follows:
Since `i1` and `i2` both are of type `int`, `i1 / i2` is an integer division which results in 0. So `0 * i2 = 0`. But `d` is of type `double` so result would be displayed as `0.0`. The part of the statement `i2 * i1` would be computed as `290` which is an `int`. Next, `290 / i2` is an integer division which would result as `10`, Finally, the result is assigned to `d`. Since `d` is of type `double`, it would be printed as `10.0`.

5. Consider the Java program below. [MCQ:2 points]

```
class FClass{
    int i;
    public static void main(String[] args) {
        final int i = 10;
        i = 20;
        System.out.println(i);
    }
}
```

Choose the correct option regarding the given program.

- √ **The program generates a compilation error.**
- ◯ The program generates a run time error.
- ◯ The program generates output: 10
- ◯ The program generates output: 20

Solution: `i` is declared as `final`. Thus, it cannot be reassigned.

6. Consider the Java program below. [MCQ:2 points]

```
class SClass{
    int i = 10;
    public SClass(int _i) { i = 20; }
    public void print() { System.out.println(i); }
}
class PQ{
    public static void main(String[] args) {
        int i = 30;
        SClass obj = new SClass(i);
        obj.print();
    }
}
```

Choose the correct option regarding the given program.

- ◯ The program generates a compilation error.
- ◯ The program generates output: 10
- √ **The program generates output: 20**
- ◯ The program generates output: 30

Solution: The constructor:
`public SClass(int _i) { i = 20; }`
initializes `i` to 20. Thus, it prints 20.

7. Consider the Java program below. [MCQ:2 points]

```
class StringExample {
    public static void main(String[] args) {
        String str = "we are here to learn java";
        String str1 = "";
        char[] ch = {‘a’, ‘e’, ‘i’,‘o’, ‘u’};
        for(char c : ch){
            for (int i = 0; i < str.length(); i++){
                if (str.charAt(i) == c){
                    str1 += c;
                }
            }
        }
    }
}
```

What will be the value of `str1` at the end of execution?

- ◯ `"eaeeeoeaaa"`
- ◯ `"wrhrtlrnjv"`
- √ `"aaaaeeeeeo"`
- ◯ `"aeiou"`

Solution:

- Program execution starts from main function
- `str1` is assigned to a string, `str2` is an empty string.
- `ch` is a character array, and is initialized to {`’a’, ‘e’, ‘i’, ‘o’, ‘u’`}
- Each element of the character array is compared with each and every element of `str`. if both these elements are same, then that element is concatenated, to `str1`.
- Hence `str1` will have all the vowels present in `str` and are arranged in alphabetical order.

8. Consider the Java code given below. [MCQ : 2 points]

```
import java.util.Scanner;
class SwitchEx
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Please enter the any ASIA cricket team");
        String country=sc.next();
        switch(country)
        {
            case "India":
                System.out.println("Delhi is capital of "+country);
            case "SriLanka":
                System.out.println("Colombo is capital of "+country);
            default:
                System.out.println("Please enter valid country name:");
                break;
        }
    }
}
```

If we provide `SriLanka` as the input to the program, what will be the output?

- ◯ Compilation fails.
- ◯ `Colombo is capital of India.`
- ◯ `Delhi is capital of India.`
  `Colombo is capital of SriLanka.`
  `Please enter valid country name:`
- √ `Colombo is capital of SriLanka`
  `Please enter valid country name:`

Solution: If you give `SriLanka` as input to the above program, it matches to the second case and executes corresponding statements, there is no break used after the case, it will execute default also.

9. Consider the Java code given below. [MCQ : 2 points]

```
class  ForComma
{
    public static void main(String[] args)
    {
        int j=20;
        for (int i=10;i<j ;i++,j--)    //LINE-1
        {
            System.out.println("Hello World!");
        }
    }
}
```

Choose the correct option regarding the given code.

- ◯ Compilation fails due to error at `LINE-1`.
- ◯ `Hello World!` printed 9 times.
- √ `Hello World!` **printed 5 times.**
- ◯ `Hello World!` printed until stack overflow.

Solution:

- First iteration `j=20, i=10, i<j => true`,
  - Thus `Hello World!` will be printed.
- Second iteration `j=19, i=11, i<j => true`
  - Thus `Hello World!` will be printed.
- Third iteration `j=18, i=12, i<j => true`,
  - Thus `Hello World!` will be printed.
- Fourth iteration `j=17, i=13, i<j => true`,
  - Thus `Hello World!` will be printed.
- Fifth iteration `j=16, i=14, i<j => true`

- Thus `Hello World!` will be printed.

- Sixth iteration `j=15, i=10, i<j => false`,
  - Thus loop will be terminated.

10. Consider the Java code given below. [MCQ : 2 points]

```
class  ForEx
{
    public static void main(String[] args)
    {
        for (;;)
        {
            System.out.println("Hello World!");
        }
    }
}
```

Choose the correct option regarding the given code.

- ◯ The code generates a compilation error.
- ◯ The code generates a run time error.
- √ `Hello World!` **would be printed forever.**
- ◯ Nothing will be printed.

Solution: No error in the code, `Hello world!` message would be printed infinite times as there is no exit condition for the given *for* loop.

11. Consider the Java code given below. [MCQ : 2 points]

```
public class Demo
{
    public static void main(String args[])
    {
        int x=10;
        int y=20;
        y=+x;
        System.out.println(y);
    }
}
```

what will be the output?

- ◯ compilation failed.
- √ 10
- ◯ 20
- ◯ 30

Solution: `10` will be printed as output.
`y=+x;` this statement will assign the `x` value to the `y`, `y=+x` and `y=x` both are same.

12. Consider the Java code below. [MCQ:2 points]

```
class FClass
{
    public static void main(String apple[])
    {
        int a = 34.0;
        int b = 7;
        int c = a % b;
        System.out.println(c);
    }
}
```

Answer the following.

(a) The above code does not compile because the name of the String array is not proper.

- ◯ True
- √ **False**

(b) The above code gives 6 as the output.

- ◯ True
- √ **False**

(c) The above code does not compile because the data types are mismatched.

- √ **True**
- ◯ False

Solution:

Inside the `main()` method a variable `a` is declared which is of type `int`. So the value that `a` can hold any integer value. But `a` is assigned to `34.0` which is of type `double`. This leads to mismatch of type. So compilation error occurs.

To make this code work we need to explicitly typecast it because the double data type is larger than int. The proper version of this code yields `6` as an output.

```
class FClass
{
    public static void main(String apple[])
    {
        int a = (int)34.0;
        int b = 7;
        int c = a % b;
        System.out.println(c);
    }
}
```

13. Answer the following. [MCQ:2 points]

(a) The following is a legal array declaration.

```
String[] State = new String[] {"Chennai", "Kolkata", "Tripura" };
```

√ **True**

◯ False

(b) Arrays are objects

√ **True**

◯ False

(c) An array index runs from `1` to `a.length`

◯ True

√ **False**

(d) Size of an array expands automatically when it is full.

◯ True

√ **False**

Solution:
An array index runs from `0` to `a.length-1`.
Size of an array can not expand automatically.

14. Considering the following Java code, choose all the correct statement/(s) from among the given options regarding this code.

[MSQ:2points]

```
class Employee
{
    int eid;
    String ename;

    public Employee(Employee e)
    {
        this.eid = e.eid;
        this.ename = e.ename;
    }
    public void display()
    {
        System.out.println("eid: "+eid);
        System.out.println("ename: "+ename);
    }
}

class Fclass
{
    public static void main(String[] args)
    {
        Employee e1 = new Employee();
        e1.display();
    }
}
```

√ **One can't create an object of the *Employee* class.**

◯ Program runs successfully, and the data members `eid` and `ename` contains random garbage value.

◯ Program runs successfully, and the data members `eid` and `ename` contains *0* and *null* respectively.

√ **This code results in a compilation error.**

Solution:
In the above code a copy constructor is defined and therefore the default no argument constructor is hidden. Now the code has neither the default constructor nor a explicitly defined constructor for creating a parameterized or non-parameterized

object. So **we can't create an object of the Employee class.**
Therefore the statement `Employee e1 = new Employee();`, creates a compile time error.

15. What will be the output of the following Java code?

[MCQ:2points]

```
import java.util.*;

class Employee
{
    int eid;
    String ename;
    String eprojects[];

    public Employee(Employee e)
    {
        this.eid = e.eid;
        this.ename = e.ename;
        this.eprojects = e.eprojects;
    }
    public Employee(int id, String name, String[] projects)
    {
        this.eid = id;
        this.ename = name;
        this.eprojects = projects;
    }
    public void display()
    {
        System.out.println("eid: "+eid);
        System.out.println("ename: "+ename);
        System.out.println("eprojetcs: ");
        for(String s : eprojects){
            System.out.println(s);
        }
    }
    public void mutator()
    {
        this.ename = this.ename + "__IITM";
        this.eprojects[0] = "P6";
    }

}
public class FClass
{
    public static void main(String[] args)
    {
```

```
        String arr[] = {"P1","P2"};
        Employee e1 = new Employee(12,"Alex",arr);
        Employee e2 = new Employee(e1);
        e1.mutator();
        e1.display();
        e2.display();
    }
}
```

- ◯ This program results in a compilation error.
- ◯ eid: 12
  ename: Alex__IITM
  eprojetcs:
  P6
  P2
  eid: 12
  ename: Alex__IITM
  eprojetcs:
  P6
  P2
- ◯ eid: 12
  ename: Alex__IITM
  eprojetcs:
  P6
  P2
  eid: 12
  ename: Alex
  eprojetcs:
  P1
  P2
- √ **eid: 12**
  ename: Alex__IITM
  eprojetcs:
  P6
  P2
  eid: 12
  ename: Alex
  eprojetcs:
  P6
  P2

Solution:
The statement `this.eprojects = e.eprojects;` in the copy constructor does a shallow copy of the eprojects array therefore when the array is mutated for object *e1* the same change is reflected for instance *e2* as well.
Will be discussed in the live session.

BSCCS2005: Practice Assignment
Questions with Solutions
Week 3

1. Consider the following code. [Nalini: MCQ: 2points]

```
public class Example1{
    public int a = 10;
    public void display( ){
        System.out.print("In parent class");
    }
}
public class Example2 extends Example1{
    public int b = 20;
    public void display( ){
        System.out.println("In child class");
    }
    public static void main(String[] args) {
        Example1 obj = new Example2( );
    }
}
```

What is the output of `obj.display( )`?

- ◯ In parent class
- √ **In child class**
- ◯ Compiler Error
- ◯ In parent class
  In child class

Solution:

- obj is a reference variable of type Example1 but is referring to an object of type Example2.
- display( )method is overridden in subclass.
- So calling obj.display( ) prints In child class.

2. Consider the following code. [Nalini: MSQ: 2points]

```
public class Polygon {
    public static void perimeter( ){
        System.out.print("In Polygon perimeter");
    }
    public void area( ) {
        System.out.println("In Polygon area");
    }
    public void sides( ){
         System.out.println("In Polygon sides");
    }
    public void angleSum( ) {
        System.out.println("In Polygon angleSum");
    }

}
public class Pentagon extends Polygon{
    public static void perimeter( ){
        System.out.println("In Pentagon perimeter");
    }
    public void area( ) {
         System.out.println("In Pentagon area");
    }
    public int sides( ){
         return 5;
    }
    public void angleSum(int x) {
        System.out.println("In Pentagon angleSum");
    }
}
```

Which method/s hide/s methods in the parentclass?

- √ **perimeter**
- ◯ area
- ◯ sides
- ◯ angleSum

Solution:

- If the static methods of same signature are present in both parentclass and subclass then static methods of parentclass are hidden in subclass

3. Consider the following code. [Nalini: MSQ: 2points]

```
1    public class Animals{
2        final public void swim( ){
3            System.out.println("Animals swim");
4        }
5        public static void communicate( ){
6            System.out.println("Animals communicate");
7        }
8    }
9    public class Birds extends Animals{
10        public void swim( ){
11            System.out.println("Birds swim");
12        }
13        public void communicate( ){
14            System.out.println("Birds communicate");
15        }
16        public void fly( ){
17            System.out.println("Birds fly");
18        }
19   }
```

Identify the line/s which has/have error.

- ◯ Line 1
- ◯ Line 2
- ◯ Line 5
- ◯ Line 9
- √ **Line 10**
- √ **Line 13**
- ◯ Line 16

Solution:

- final methods cannot be overridden
- static methods cannot be overridden

4. Identify the steps in implementation of:
   A. structured programming
   and
   B. object-oriented programming.

   1. Identify the data to be manipulated and maintained
   2. Design a set of procedures for specific tasks and combine them to build a complex system
   3. Devise / identify algorithms to operate on the data
   4. Design data structures to suit procedural manipulations

[ARUP: MCQ: 2 points]

- ◯ A. $1 \rightarrow 4$
  B. $2 \rightarrow 3$
- ◯ A. $1 \rightarrow 3$
  B. $2 \rightarrow 4$
- ◯ A. $1 \rightarrow 2$
  B. $3 \rightarrow 4$
- √ **A. $2 \rightarrow 4$**
  B. $1 \rightarrow 3$

Solution: In structured programming,

- First, design a set of procedures for specific tasks and combine them to build a complex system
- Then, design data structures to suit procedural manipulations

In object oriented programming,

- First, identify the data need to be maintained and manipulated
- Then, identify algorithms to operate on the data

5. Consider the Java code given below. [ARUP: MSQ: 2 points]

```
public class Parent{
    public void f1(){
        System.out.println("f1() called from parent class");
    }
}
public class Child extends Parent{
    public void f1(){
        System.out.println("f1() called from child class");
    }
}
public class FClass{
    public static void main(String[] args){
        //LINE 1
    }
}
```

Which of the following statements at `LINE 1` will generate the output as:
`f1() called from parent class`

- √ `new Parent().f1();`
- ◯ `((Child)new Parent()).f1();`
- ◯ `new Child().f1();`
- ◯ `((Parent)new Child()).f1();`

Solution:

- In option-1, `new Parent()` creates an unnamed object of `Parent` type, so `f1()` will be called from `Parent` class. Thus, output is `f1() called from parent class`.
- In option-2, `new Parent()` creates an unnamed object of `Parent` type. However, type casting from a `Parent` type to `Child` type (subtype) is illegal. Thus, it generates compiler error.
- In option-3, `new Child()` creates an unnamed object of `Child` type, so `f1()` will be called from `Child` class. Thus, output is `f1() called from child class`.
- In option-4, `new Child()` creates an unnamed object of `Child` type, which is type casted to `Parent` type. Although, the static type of the object is `Parent`, dynamic type is `Child`. Thus, dynamic dispatch calls the `f1()` from `Child` class. Thus, the output is `f1() called from child class`.

6. Consider the Java code given below. [ARUP: MCQ: 2 points]

```
public class Rectangle{
    private final void computeArea(){
        System.out.println("area of rectangle");
    }
}
public class Cube extends Rectangle{
    public final void computeArea(){
        System.out.println("area of cube");
    }
}
public class FClass{
    public static void main(String[] args){
        Cube r = new Cube();
        r.computeArea();
    }
}
```

What will be the output?

- ◯ `area of rectangle`
  `area of cube`
- ◯ `area of rectangle`
- √ `area of cube`
- ◯ It generates compiler error

Solution: Although `final` keyword makes a method not overridable, the `computeArea()` in `Rectangle` is invisible outside of the class. Thus, when the method redefined `computeArea()` in `Cube` it would not be considered as overriding, rather it is considered as a new method defined in `Cube`. Thus, it prints `area of cube`.

7. Consider the code segment given below. [ARUP: MSQ: 2 points]

```
public class Data{
    //definition of MAX as constant
}
public class FClass{
    public static void main(String[] arg){
        System.out.println(Data.MAX);
    }
}
```

Consider that `MAX` is defined as constant in `class Data`. Identify the most appropriate definition(s) of `MAX` in `class Data` such that the output of the code is `100`.

- ◯ `public final int MAX = 100;`
- ◯ `private final static int MAX = 100;`
- ◯ `private static int MAX = 100;`
- √ `public final static int MAX = 100;`

Solution: Since `MAX` is accessed using the class name, it must defined as `static`. Since we have considered that `MAX` is a constant, , it must defined as `final`. Since `MAX` is accessed from the outside of the class where it is defined, it must be defined with the default or `public` access specifier.

8. Consider the Java code given below. [ARUP: MCQ: 2 points]

```
public class Employee{
    private int empid;
    private String name;

    public Employee(int empid_, String name_){
        empid = empid_;
        name = name_;
    }
    public Employee(){
        this(0, "unknown");
    }
    public void print()
        System.out.print(empid + " : " + name + " : ");
    }
}
public class Manager extends Employee{
    private String department;

    public Manager(int empid_, String name_, String department_){
        department = department_;
    }
    public void print(){
        super.print();
        System.out.println(department);
    }
}
public class FClass{
    public static void main(String[] args){
        Manager m = new Manager(101, "Nutan", "HR");
        m.print();
    }
}
```

What will be the output?

- ◯ `0 : unknown`
- √ `0 : unknown : HR`
- ◯ `HR`
- ◯ `101 : Nutan : HR`

9. Consider the following code. [Bhaskar:MCQ:2points]

```
public class A{
    public float g;

    public A(){
        g = 9.8f;
    }
    public void show(){
        System.out.println("g = "+g);
    }
}
public class B extends A{
    public double e;

    public B(double num){
        e = num;
    }
    public void show(){
        System.out.println("e = "+e);
    }
}
public class FClass{
    public static void main(String args[]){
        A obj1 = new B(2.718);

        A ptr1 = (A)obj1;
        A ptr2 = (B)obj1;
        B ptr3 = (B)obj1;

        ptr1.show();
        ptr2.show();
        ptr3.show();
    }
}
```

What will be the output of the given code?

- ◯ g = 9.8
  g = 9.8
  e = 2.718
- ◯ g = 9.8
  e = 2.718
  e = 2.718

√ **e = 2.718**
e = 2.718
e = 2.718

◯ g = 9.8
g = 9.8
g = 9.8

Solution:
The method `show()` is overridden in the child class therefore decision regarding which version of `show()` is to be executed is done at runtime depending upon the instance/object type (and not on reference type).
`(A)obj1`, `(B)obj1` is casting the type of reference not object, the object's type is *B* in all the cases and thus `show()` method of *class B* is invoked in each of them.

10. Consider the following code. [Bhaskar:MCQ:2points]

```
1. public class Mammal{
2.     public String name;
3.     public int lifespan;
4.
5.     public Mammal(){
6.         name = "Giraffe";
7.         lifespan = 45;
8.     }
9.
10.    public void show(){
11.         System.out.println("Giraffe");
12.         System.out.format("name = %s : lifespan = %d",name,lifespan);
13.    }
14. }
15.
16. public class Endangered extends Mammal{
17.     public boolean endanger_status;
18.
19.     public Endangered(){
20.         endanger_status = false;
21.     }
22.
23.     public void show(){
24.         System.out.println("endanger status of "+this.name
                                + " is "+endanger_status);
25.     }
26.     public void display(){
27.         this.show();
28.     }
29. }
30.
31. public class FClass{
32.     public static void main(String args[]) {
33.         Mammal m1 = new Endangered();
34.         m1.show();
35.         m1.display();
36     }
37. }
```

Choose the correct option regarding the given program.

- ◯ The program generates output:
  name = Giraffe : lifespan = 45
  endanger status of Giraffe is false
- ◯ The program generates output:
  endanger status of Giraffe is false
  endanger status of Giraffe is false
- ◯ The program produces a compilation error on `LINE 34`
- √ **The program produces a compilation error on** `LINE 35`
- ◯ The program produces a compilation error on `LINE 23`

Solution:

In this example the `show() method of Mammal class is overridden in the Endangered class but display() method is a method of Endangered class only. Therefore display() method will always be invoked depending upon the reference type and hence it can be invoked by a reference of Endangered type.`
`But here m1 is a reference of Mammal type and hence the statement m1.display() produces a compilation error.`

11. What will be the output of the following Java program?

[Bikash:MCQ:2 points:Lecture 3.2]

```
public class Numbers{
    public int x = 1;
    public double y = 2.1;

    public Numbers(int a, double b){
        add(a,a);
        add(b,a);
    }
    public void add(int a, int b){
        this.x = a + b;
    }
    public void add(double c, int d){
        this.y = c + d;
    }
}
public class Example{
    public static void main(String args[]){
        Numbers obj = new Numbers(2,3.2);
        System.out.println(obj.x + " " + obj.y);
    }
}
```

- ◯ Compilation failed.
- ◯ 1 2.1
- ◯ 4 6.4
- √ **4 5.2**

Solution: The steps of execution of the above program is:

- We have a parametric constructor that accepts an `int` and a `double`.
- This constructor calls the 2 overloaded methods `add`, one after another, depending upon the passed arguments.
- These methods changes the values of x and y respectively and we get the desired output.

12. Consider the Java program given below and select the correct statement.

[Anand : MCQ : 2 points]

```
public class A{
    private int a=10;
    private int b=20;
}
public class B extends A{
}
public class Example{
    public static void main(String[] args){
        B ob=new B();
        A oa=new A();
        ob.a=20;     // Line 1
        System.out.println(oa.a);  // Line 2
        System.out.println(ob.a);  // Line 3
    }
}
```

- ◯ Illegal access of variable `a` at Line 1.
- ◯ Illegal access of variable `a` at Line 2.
- ◯ Illegal access of variable `a` at Line 3.
- √ **All of the above.**

Solution: Line 1 it's not possible to change the private instance variable of super class directly.
Line 2 it's not possible to access/change the private instance variable outside the class.
Line 3 it's not possible to access the private instance variable of super class directly.

13. Consider the statements given below and select the correct option.

[Anand : MCQ : 2 points]

Statement 1: If the same static method exists in both super class and subclass, then the subclass method hides the super class method.
Statement 2: Both the variables and the methods in a super class can be overridden in a subclass.

- ◯ Both statements are true.
- ◯ Both statements are false.
- √ **Statement 1 is true and Statement 2 is false.**
- ◯ Statement 1 is false and Statement 2 is true.

Solution: Static methods of super class can be hidden by the subclass methods; overriding of static methods in subclass is not possible.
Methods are polymorphic, we can override the methods in subclass of parent class.
Variables are not polymorphic, we cannot override the variables in subclass of parent class.

14. Consider the Java program given below and predict the output.

[Anand : MCQ : 2 points]

```
public class A{
    public static void print(){
        System.out.println("Class A print");
    }
    public void show(){
        System.out.println("Class A show");
   }
}
public class B extends A{
    public static void print(){
        System.out.println("Class B print");
    }
   public void show(){
        System.out.println("Class B show");
   }
}
public class MyClass{
    public static void main(String args[]){
        A oa=new B();
        oa.print();
        oa.show();
    }
}
```

- √ **super class print**
  sub class show
- ◯ super class print
  super class print
- ◯ sub class show
  sub class show
- ◯ Compilation failed.

Solution: print() method is overridden in the subclass because it is a static method. show() overridden in the subclass because it is non-static method.

15. What will be the output of the following Java program?

[Bikash:MCQ:2 points:Lecture 3.3]

```
public class College{
    public void name(){
        System.out.println("IIT Madras.");
    }
}
public class Example extends College{
    public void name(){
        super.name();
        System.out.println("University of Calcutta.");
    }
    public static void main(String[] args){
        Example obj = new College();
        obj.name();
    }
}
```

√ **Compilation fails.**

◯ IIT Madras.
University of Calcutta.

◯ IIT Madras.

◯ University of Calcutta.

Solution:
The reference variable of the Parent class is capable of holding its object reference as well as its child object reference. But the reference variable of the child class can only hold its object reference. Hence, we can create an object of the parent class by taking reference of the child class.

Consider the Java program given below and answer the questions 16 to 18.

[Bikash:MCQ:2 points:Lecture 3.3]

```
public class A{
    public String features(){
        return "Write Once, Run Anywhere.";
    }
}
public class B extends A{
    public String features(){
        return "Object Oriented.";
    }
}
public class C extends A{
    public String features(){
        return "Multithreaded.";
    }
}
public class Example{
    public static void main(String args[]){
        A obj1 = new A();
        A obj2 = new B();
        A obj3 = new C();
        System.out.print(obj1.features()+"\n"
        +obj2.features()+"\n"+obj3.features());
    }
}
```

16. The above program is an example of

    √ **Dynamic dispatch.**

    ◯ Compile time polymorphism.

17. The output of the above program is:

```
Write Once, Run Anywhere.
Write Once, Run Anywhere.
Write Once, Run Anywhere.
```

    ◯ True

    √ **False**

18. What is the return type of the overridden method?

    ◯ void

√ **String**

Solution:
(a) The above program is a sheer implementation of dynamic dispatch. Based on the object being referred, the appropriate overridden method will be called at the run time.

(b) The output of the above program is:

```
Write Once, Run Anywhere
Object Oriented.
Multithreaded.
```

19. Consider the Java program given below and predict the output.

[Anand : MCQ : 2 points]

```
public class A{
    public void show(){
        System.out.println("A class show()");
    }
}
public class B extends A{
    public void show(){
         System.out.println("B class show()");
    }
}
public class C extends B{
}
public class  Example{
    public static void main(String[] args){
        C oc=new C();
        oc.show();
    }
}
```

- ◯ Compilation failed
- ◯ Nothing will be printed
- ◯ A class show()
- √ **B class show()**

Solution: oc.show();
The above statement will first look into class C for show(). Since it does not find show() method in C, it will look into its base class B. If show() was not available in B, then it would look inside class A and so on.
show() available in B, therefore control will execute the show() from class B.

20. Consider the Java program given below and predict the output.

[Anand : MCQ : 2 points]

```
public class A{
    public A(){
        System.out.println("A() called");
    }
}
public class B extends A{
    public B(){
        System.out.println("B() called");
    }
}
public class C extends B{
    public C(){
        System.out.println("C() called");
    }
}
public class  Example{
    public static void main(String[] args){
        C oc=new C();
    }
}
```

- ◯ A() called
- ◯ C() called
- ◯ B() called
  C() called
- √ **A() called**
  B() called
  C() called

Solution: Whenever objects are created for subclass automatically object is created for base class also, whenever object created for class C its base class B also gets instantiated, Here class B has its parent class A also gets instantiated.
All constructors can be called in above program.

21. What will be the output of the following Java program?

[Bikash:MCQ:2 points:Lecture 3.6]

```
public final class Music{
    public void artist(){
        System.out.println("Anupam Datta");
    }
}
public class Poet extends Music{
    public void artist(){
        super.artist();
        System.out.println("Rabindranath Das");
    }
}
public class Example{
    public static void main(String args[]){
        Music obj = new Poet();
        obj.artist();
    }
}
```

- ◯ Anupam Datta
- ◯ Rabindranath Das
- ◯ Anupam Datta
  Rabindranath Das
- √ **Compile time error.**

Solution:
The class `Music` has been declared `final`. Hence, it cannot be inherited by any other class. The compiler will immediately throw an error after detecting that we are trying to extend a `final` class.

BSCCS2005: Practice Assignment with Solutions
Week 4

1. Consider the code given below. [MCQ:2 points]

```
public interface Shape{
    public double area();
    public default double volume() {
        return -1.0;
    }
}

public interface Printable{
    public default void print() {
        System.out.println("not implemented");
    }
}

public class Rectangle implements Shape, Printable{
    private double w, h;
    public Rectangle(double w_, double h_) {
        w = w_;
        h = h_;
    }
    public double area() {
        return w * h;
    }
    public void print() {
        System.out.print(area() + " ");
        System.out.print(volume());
    }
}

public class FClass{
    public static void main(String[] args) {
        Rectangle r = new Rectangle(20.0, 50.0);
        r.print();
    }
}
```

What will be the output?

- ◯ `1000.0` followed by a runtime error
- ◯ `not implemented`
- √ `1000.0 -1.0`
- ◯ It generates compiler error

Solution: In `Rectangle` class, the method `print()` from `Printable` and the method `area()` from `Shape` are overridden. Since, the object is of `Rectangle` type, the methods `print()` and `area()` invoke the implementations from `Rectangle` class. However, since `volume()` is not overridden, the default implementation will be inherited. Thus, the output is `1000.0 -1.0`.

2. Consider the code given below. [MCQ:2 points]

```
public interface Flyable{
    public void fly();
    public default void travel() {
        System.out.println("travel with wings");
    }
}

public interface Movable{
    public void move();
    public default void travel() {
        System.out.println("travel with legs");
    }
}

public class Bird implements Flyable, Movable{
    public void fly() {
        System.out.println("fly with wings");
    }
    public void move() {
        System.out.println("move with legs");
    }
}

public class FClass{
    public static void main(String[] args) {
        Bird b = new Bird();
        b.fly();
        b.move();
        b.travel();
    }
}
```

What will be the output?

- ◯ `fly with wings`
  `move with legs`
- ◯ `fly with wings`
  `move with legs`
  `travel with wings`
- ◯ `fly with wings`
  `move with legs`
  `travel with legs`

√ It generates a compiler error

Solution: Call to the `travel()` method in `class Bird` is ambiguous, since it inherits two different versions of default implementations from two different interfaces.

3. Consider the code given below. [MCQ:2 points]

```
public interface Comparable{
    public abstract int comp(Comparable x);
}
public class Name implements Comparable{
    private String fname;
    private String lname;

    public Name(String fname, String lname){
        this.fname = fname;
        this.lname = lname;
    }
    public int comp(Comparable x) {
        if(lname.compareTo(((Name)x).lname) == 0)
            return fname.compareTo(((Name)x).fname);
        return lname.compareTo(((Name)x).lname);
    }
    public void print() {
        System.out.println(fname + " " + lname);
    }
}
public class FClass{
    public static void sort(Comparable[] names) {
        for(int i = 0; i < names.length - 1; i++) {
            for(int j = 0; j < names.length - i - 1; j++) {
                if(names[j].comp(names[j + 1]) > 0) {
                    Comparable tname = names[j];
                    names[j] = names[j + 1];
                    names[j + 1] = tname;
                }
            }
        }
    }
    public static void main(String[] args) {
        Name[] names = new Name[] {new Name("Charlotte", "Brown"),
                                   new Name("Ava", "Smith"),
                                   new Name("Emma", "Williams"),
                                   new Name("Olivia", "Smith"),
                                   new Name("Emma", "Johnson")};
        sort(names);
        for(int i = 0; i < names.length; i++)
            names[i].print();
    }
```

```
}
```

What will be the output?

- ◯ `Ava Smith`
  `Charlotte Brown`
  `Emma Johnson`
  `Emma Williams`
  `Olivia Smith`
- √ `Charlotte Brown`
  `Emma Johnson`
  `Ava Smith`
  `Olivia Smith`
  `Emma Williams`
- ◯ `Ava Smith`
  `Charlotte Brown`
  `Emma Williams`
  `Emma Johnson`
  `Olivia Smith`
- ◯ It generates an error

Solution: As per the `comp()` implementation in `Name` class, the names will be first sort by `lname`, and in case of equality of `lname`, sort by `fname`.

4. Consider the code given below. [MSQ:2 points]

```
1 public abstract class Polygon{
2   public abstract int perimeter();
3 }
4 public interface Shape{
5   public abstract int area();
6 }
7 public class Rectangle extends Shape implements Polygon{
8   public int area( ){
9     System.out.println("length*breadth");
10  }
11   public int perimeter( ){
12     System.out.println("length + breadth");
13   }
14 }
```

Identify the error in the code.

- ◯ Line 2: perimeter( ) is not defined
- ◯ Line 5: area( ) is not defined
- √ Line 7: incorrect usage of extends and implements
- ◯ Line 7: class Rectangle is not declared as abstract
- ◯ Line 8: area( ) has taken arguments
- √ Line 10: missing return statement
- √ Line 13: missing return statement

Solution:

- Line 7: public class Rectangle extends Polygon implements Shape
- area(), perimeter() should return int

5. Consider the program given below and choose the correct option.

[MCQ:2 points]

```
public abstract class NewYear {
    abstract String resolution();
    public NewYear(){
        System.out.println("Resolution: ");
    }
}
public class Year_2022 extends NewYear{
   public String resolution() {
      return "Walk up early, exercise and take shower everyday.";
   }
   public static void main(String args[]) {
      System.out.print(new Year_2022().resolution());
   }
}
```

- ◯ We cannot declare a constructor of the `abstract` class.
- ◯ Compile time error.
- √ Output of this program is:
  Resolution:
  Walk up early, exercise and take shower everyday.
- ◯ Output of this program is:
  Walk up early, exercise and take shower everyday.

Solution: The compiler adds a constructor to an `abstract` class if a programmer does not declare it explicitly. Constructor of an abstract class is necessary because the subclass constructors calls the base class constructor by default.

6. Consider the program given below and choose the correct option.

[MCQ:2 points]

```
public interface Rectangle{
   abstract int areaRectangle(int length,int breadth);
}
public interface Circle{
   abstract int areaCircle(int radius);
}
public interface Shape extends Rectangle,Circle{

}
public class TwoDimension implements Shape{
   public int areaRectangle(int length,int breadth){
      return length*breadth;
   }
   public int areaCircle(int radius){
      return (int)(Math.PI*Math.pow(radius,2));
   }
}
public class Example{
   public static void main(String[] args){
      TwoDimension ref=new TwoDimension();
      System.out.println("The area of the rectangle is: "+ref.areaRectangle(5,2)
      +"\n"+"The area of the Circle is: "+ref.areaCircle(5));
   }
}
```

- ◯ This program generates a compile time error because an interface cannot be extended.
- ◯ The area of the rectangle is: 10 followed by a runtime error.
- √ The area of the rectangle is: 10
  The area of the Circle is: 78
- ◯ The area of the rectangle is: 10
  The area of the Circle is: 75

Solution: Classes implement interfaces, and interfaces are allowed to inherit from other interfaces.

7. Consider the Java program given below and predict the output. [MCQ : 2 points]

```
public abstract class Base{
    public Base(){
        System.out.println("Base class constructor");
    }
}
public class Sub extends Base{
    public Sub(){
        System.out.println("Subclass constructor");
    }
}
public class Example {
    public static void main(String[] args){
        Base base=new Sub();
    }
}
```

- ◯ Compilation failed, you cannot define a constructor for abstract class.
- ◯ Compilation failed because of the statement below.
  Base base=new Sub();
- √ Base class constructor
  Subclass constructor
- ◯ Run time error

Solution: Option 3
You can define constructor in the abstract class. It executes when its subclass gets instantiated.

8. Consider the Java program given below and choose the correct option.

[MCQ : 2 points]

```
public interface City{
    public abstract void travel(String name);
}
public class Mumbai implements City{
    public void travel(String name) {
        System.out.println(name+" Travelling to Mumbai");
    }
}
public class Hyderabad implements City{
    public void travel(String name) {
        System.out.println(name+" Travelling to Hyderabad");
    }
}
public class Traveller{
    private City city;
    private String name;
    public Traveller(City city,String name) {
        this.city = city;
        this.name = name;
    }
    public void journey(){
        city.travel(name);
    }

}
public class Example{
    public static void main(String[] args) {
        Traveller traveller1=new Traveller(new Hyderabad(),"Johny");
        traveller1.journey();
        Traveller traveller2=new Traveller(new Mumbai(),"Virat");
        traveller2.journey();
    }
}
```

- ◯ Compilation failed because an interface cannot be a member of another class.
- ◯ Compilation failed because Traveller constructor will not accept Hyderabad and Mumbai class objects.
- √ Johny Travelling to Hyderabad
  Virat Travelling to Mumbai
- ◯ null Travelling to null
  null Travelling to null

Solution: interface can hold its implemented classes objects.
`traveller1.journey();` generates `Johny Travelling to Hyderabad` as output.
`traveller2.journey();` generates `Virat Travelling to Mumbai` as output.

9. Choose all the correct option(s) regarding the code given below.

```
public interface Academics{
    default void getName(){
        System.out.println("Academics : getName");
    }
    default void printName(){
        System.out.println("Academics : printName");
    }
}

public class University{
    private int univ_id;

    private class College implements Academics{
        private String college_name;

        public void getName(){
            System.out.println(college_name);
        }

        public void getID(){
            System.out.println(univ_id);
        }

        College(String name){
            college_name = name;
        }
    }
    public College getReference(){
        return new College("IITMadras");
    }
}
public class FClass{
        public static void main(String[] args) {
            University uni = new University();
            Academics acad = uni.getReference();
            acad.getName();
            acad.getID();
        }
}
```

[MSQ:2points]

- ◯ This code generates output:
  IITMadras 0
- ◯ This code generates compilation error because default method `getName` is overridden.
- ◯ This code generates compilation error because method `getReference` does not return a valid `College` type object.
- ◯ This code generates compilation error because private instance variable `univ_id` of outer class can not be directly accessed inside method `getID` of inner class.
- √ This code generates compilation error because `getID` method of class `College` is not accessible by `acad` object..

10. Consider the Java program given below.

[MCQ : 2 points]

```
import java.util.Date;
public class Timer{
    private Employee e;
    public Timer(Employee e){
        this.e=e;
    }
    public void start(Date date){
        if(date.getDay()==6 || date.getDay()==0){
            e.notification1();
        }
        else{
            e.notification2();
        }
    }
}
public class Employee {
    public void check(){
        Timer obj=new Timer(this);
        Date d=new Date();
        obj.start(d);
    }
    public void notification1(){
        System.out.println("Happy weekend...");
    }
    public void notification2(){
        System.out.println("Today is a working day, stay at work.");
    }
    public static void main(String[] args) {
        Employee e=new Employee();
        e.check();
    }
}
```

Note that the `getDay()` method returns an integer value from 0 to 6. Each integer represents a day of the week as follows:
0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday.
If this program in run on a Sunday, what will be the output?

- ◯ This code generates output: `Today is a working day, stay at work.`
- ◯ This code generates a compile time error.
- ◯ This code generates no output.

√ This code generates the output: Happy weekend...

Solution: Assume if you execute this program on Sunday, getDay() method will return 0.
If getDay() is equal to 0 or 6 it will call notification1(), otherwise notification2().

BSCCS2005: Practice Assignment with Solutions
Week 5

1. Consider the code given below. [MCQ:2 points]

```
public abstract class OutputDevice{
    public abstract void output();
}
public class Printer extends OutputDevice{
    public void output() {
         System.out.println("printer prints");
    }
}
public class Monitor extends OutputDevice{
    public void output() {
        System.out.println("monitor displays");
    }
}
public interface Iterable{
    public boolean has_next();
    public Object get_next();
}
public class OutputList implements Iterable{
    private final int max_limit = 2;
    private int indx;
    private Object[] oArr = {new Printer(), new Monitor()};
    public OutputList(){
        indx = -1;
    }
    public boolean has_next() {
        if (indx < max_limit - 1)
            return true;
        return false;
    }
    public Object get_next() {
        indx++;
        return oArr[indx];
    }
}
public class FClass{
    public static void main(String[] args) {
        OutputList list = new OutputList();
        while(list.has_next()) {
            //LINE1
        }
    }
}
```

Identify the appropriate option to fill in the blank at `LINE1` such that the output is

```
printer prints
monitor displays
```

- ◯ `(list.get_next()).output();`
- ◯ `((Monitor)list.get_next()).output();`
- √ `((OutputDevice)list.get_next()).output();`
- ◯ `((OutputList)list.get_next()).output();`

Solution: `Object` type elements required to be type casted to `OutputDevice` type in order to access the `output` method.

2. Consider the code given below. Choose the correct option regarding the given code.

[Bikash:MCQ:2 points]

```
public class Example<T>{
   T ob;
   Example(T x){
      this.ob=x;
   }
   public String show(){
      return ""+ob.getClass().getName();
   }
   public T get(){
      return ob;
   }
}
public class Test {
    public static void main(String[] args){
       Example<Number> n=new Example<Number>(100);
       Example<Double> e=new Example<Double>(10.5);
       n=e;
       System.out.print(n.show()+"\n"+n.get());
    }
}
```

- √ This program generates compile time error.
- ◯ This program generates runtime error.
- ◯ This program generates output:
  java.lang.Number
  100
- ◯ This program generates output:
  java.lang.Double
  10.5

Solution: This program generates compile time error because polymorphism is not applicable for generic type arguments.

3. Consider the Java code given below, and choose the correct option/s suitable for `Line 1`. [MSQ : 2 points]

```
public class Show{
    public <T> void display(T[] elements) {
         for (T element : elements){
            System.out.println(element);
         }
         System.out.println();
    }
}
public class Example{
   public static void main(String args[]) {
        ----------------------------- //Line 1
        Show obj1=new Show();
        obj1.display(arr1);
    }
}
```

- ✓ `Integer[] arr1 = {10, 20, 30, 40, 50};`
- ✓ `String[] arr1 = {"IIT","Madras","Java","Programming"};`
- ◯ `int[] arr1 = { 10, 20, 30, 40, 50 };`
- ◯ `double[] arr1= {20.5,30.7,56.6,67.8,0.25};`

Solution: The type variable `<T>` allows any type of array object.
Option 1: `Integer` object array can be passed to the type variable `<T>`.
Option 2: `String` object array can be passed to the type variable `<T>`.
Option 3: `int` primitive type array passed to the type variable `<T>` is invalid.
Option 4: `double` primitive type array passed to the type variable `<T>` is invalid.

4. Consider the code given below and choose the correct option.

[MCQ : 2 points]

```
public class Faculty {
    private  String name;
    private  String dept;
    public String getName() {
        return name;
    }
    public String getDept() {
        return dept;
    }
    public Faculty(String name, String dept) {
        this.name = name;
        this.dept = dept;
    }
    public String toString() {
        return "Faculty [name=" + name + ", dept=" + dept + "]";
    }
}
public class Hod extends Faculty {
    public Hod(String name, String dept) {
        super(name, dept);
    }
    public String toString() {
        return "Hod [name=" + getName() + ", dept=" + getDept() + "]";
    }
}
public class CopyArrayObjects {
    public static <S extends T,T> void copy (S[] src,T[] tgt){
        int i,limit;
        limit = Math.min(src.length,tgt.length);
        for (i = 0; i < limit; i++){
            tgt[i] = src[i];
        }
    }
    public static void main(String[] args) {
        Hod hod1 = new Hod("Johny", "CSE");
        Hod hod2 = new Hod("Jock", "EEE");
        Hod hod3 = new Hod("Nelson", "CE");
        Hod hod[] = {hod1,hod2,hod3};
        Faculty[] members = new Faculty[2];
        CopyArrayObjects.copy(hod, members);
        for (int i = 0; i < members.length; i++) {
```

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

- √ This program generates output:
  `Hod [name=Johny, dept=CSE]`
  `Hod [name=Jock, dept=EEE]`
- ◯ This program generates output:
  `Faculty [name=Johny, dept=CSE]`
  `Faculty [name=Jock, dept=EEE]`
- ◯ This program generates output:
  `Hod [name=Johny, dept=CSE]`
  `Hod [name=Jock, dept=EEE]`
  `Hod [name=Nelson, dept=CE]`
- ◯ This program generates output:
  `Faculty [name=Johny, dept=CSE]`
  `Faculty [name=Jock, dept=EEE]`
  `Faculty [name=Nelson, dept=CE]`
- ◯ This code generates a compile time error.

Solution: While copying the arrays, the source array should be a subtype of the target array.

5. Consider the code given below. Choose the correct option regarding the given code.

[MCQ:2 points]

```
public interface X{
   public abstract void display();
}
public class A{
   void show(){
      System.out.println("Show");
   }
}
public class B extends A implements X{
   public void display(){
      System.out.println("Display");
   }
}
public class Example<T extends A & X>{
   T obj;
   Example(T obj){
      this.obj=obj;
   }
   void show(){
      obj.display();
   }
}
public class Main{
   public static void main(String[] args){
      Example<B> c=new Example<B>(new B());
      c.show();
   }
}
```

- ◯ This program generates output:
  Show
- √ This program generates output:
  Display
- ◯ This program generates output:
  Show
  Display
- ◯ This program generates compile time error.

Solution: If A is a class and X is an interface than <T extends A & X> means that the type variable T can take any type arguments which is child class of A and implements interface x.

6. Consider the code given below. [MCQ:2 points]

```
public class NumberFunction{
    public static <T extends Number> T max(T[] tArr){
        T max = tArr[0];
        for(int i = 0; i < tArr.length; i++) {
            if(tArr[i].doubleValue() > max.doubleValue()) {
                max = tArr[i];
            }
        }
        return max;
    }
}

public class FClass{
    public static void main(String[] args) {
        //LINE 1
        System.out.println(NumberFunction.max(arr));
    }
}
```

Identify the correct definition(s) for array `arr` for which the call `NumberFunction.max(arr)` can return the maximum element from the array `arr`.

- √ `Integer[] arr = {2, 4, 1, 6, 3};`
- √ `Double[] arr = {2.3, 4.2, 1.4, 2.6, 1.3};`
- ◯ `Character[] arr = {'H', 'e', 'L', 'l', 'o'};`
- ◯ `String[] arr = {"Apple", "test", "Apple", "Mango", "Orange"};`

Solution: For function `NumberFunction.max(arr)`, the type `T` is bounded by `Number`. Since `Integer` and `Double` both inherit from `Number`, function `NumberFunction.max(arr)` works correctly on them. However, it does not work for `Character` and `String`.

7. Consider the code given below. [MCQ:2 points]

```
public interface Verifiable{
    public abstract boolean isEqual(Object d);
}
public class Employee implements Verifiable{
    private int id;
    private String name;
    public Employee(int id, String name) {
        this.id = id;
        this.name = name;
    }
    public int get_id() {
        return id;
    }
    public String get_name() {
        return name;
    }
    public boolean isEqual(Object d) {
        if(d instanceof Employee)
            if(this.id == ((Employee)d).id)
                return true;
        return false;
    }
}
public class Manager extends Employee{
    private String department;
    public Manager(int id, String name, String department) {
        super(id, name);
        this.department = department;
    }
    public String get_department() {
        return department;
    }
}
public class FClass{
    //LINE 1: function-header
    {
        for(int i = 0; i < arr.length; i++) {
            if(m.isEqual(arr[i]))
                return true;
        }
        return false;
    }
```

```
    public static void main(String[] args) {
        Employee[] emps = {new Employee(101, "Darpan"),
            new Employee(102, "Aanya"), new Employee(103, "Binita"),
            new Employee(104, "Jairaj"), new Employee(105, "Ishaan")};
        Manager m = new Manager(103, "Binita", "IT");
        System.out.println(findEmployee(emps, m));
    }
}
```

Identify the appropriate function header for function `findEmployee(T[] arr, S m)`, such that the output is `true`

- √ `public static <T extends Verifiable, S extends T> boolean findEmployee(T[] arr, S m)`
- ◯ `public static <T, S> boolean findEmployee(T[] arr, S m)`
- ◯ `public static <T, S extends T> boolean findEmployee(T[] arr, S m)`
- ◯ `public static <T extends Verifiable, T extends S> boolean findEmployee(T[] arr, S m)`

Solution: Since `Employee` implements `Verifiable` and `Manager` extends `Employee`, relation between `T` and `S` is:
`T extends Verifiable` and `S extends T`.

8. Consider the following code. [MSQ : 2points]

```
public class Example<T extends Number>{
  private T[] arr;
  public Example(T[] a){
    arr = a;
  }
}
public class ArrayObject {
  public static void main(String[] args) {
       -------Line 1--------
  }
}
```

Choose the correct option to fill in Line 1 to create an object of Array

- ◯ `Example<Integer> a = new Example<Integer>({1,2,3,4});`
- ◯ `int[] x = {1,2,3,4};`
  `Example<Integer> a = new Example<Integer>(x);`
- √ `Integer[] x = {1,2,3,4};`
  `Example<Integer> a = new Example<Integer>(x);`
- ◯ `Example<Integer> a = new Example<Integer>( ){1,2,3,4};`
- ◯ `Example<String> a = new Example<String>({"one", "two", "three", "four"});`

Solution: `T` extends Number. So only Integer, Float, Double are compatible with `T`.
Option 1 is incorrect because of type mismatch.
Option 4 is incorrect because no constructor with zero arguments.

Consider the class `SampleClass` in the Java code given below, and answer the questions 9 and 10.

```
import java.lang.reflect.*;
public class SampleClass{
    private final int pr_data = 9;
    private String pr_str;
    public static int pu_data;
    private SampleClass() {
        //some code
    }
    public SampleClass(int pr_data_, String pr_str_) {
        pr_str = pr_str_;
    }
    public SampleClass(SampleClass tObj) {
        this.pr_str = tObj.pr_str;
    }
    private boolean isValid() {
        //some code
        return true;
    }
    public int get_pr_data() {
        return pr_data;
    }
    public String get_pr_str() {
        return pr_str;
    }
}
```

9. What should be the statements in `Line 1` and `Line 2`, respectively, such that the succeeding `for` loop prints the types of all the parameters of all the declared constructors in `SampleClass`?

```
public class FClass{
    public static void main(String[] args) {
        Class c = Class.forName("SampleClass");
        ________________________________________ //Line 1
        for(Constructor cont : my_const) {
            ________________________________________ //Line 2
            for(Class param : params) {
                System.out.print(param.getName() + ", ");
            }
        }
    }
}
```

Choose the correct option from below.

- ◯ `Constructor[] my_const = c.getConstructors();`
  `Class params = cont.Name();`
- ◯ `Constructors[] my_const = c.getAllConstructors();`
  `Class params[] = cont.getParameterTypes();`
- ◯ `Constructor[] my_const = c.getDeclaredConstructors();`
  `Class param = cont.getMethodParameters();`
- √ `Constructor[] my_const = c.getDeclaredConstructors();`
  `Class params[] = cont.getParameterTypes();`
- ◯ `Constructors[] my_const = c.getMehods();`
  `Class params[] = cont.Name();`

Solution: The solution follows from the syntax of the method in the class Class to obtain the declared constructors and the type of their parameters, of a given class.

10. If we have to print only the private instance variables of class SampleClass, which code snippet should we use?

○
```
Field[] fields1 = c.getFields();
Field[] fields2 = c.getDeclaredFields();
for(Field f2 : fields2) {
    bool = Arrays.asList(fields1).contains(f2); //fields1 contains f2?
    if (bool == true) {
        pvt_fields[i] = f2.getName(); //add to private variables
        i = i + 1;
    }
}
for (j = 0; j < i; j++) {
    System.out.println(pvt_fields[j]);
}
```

○
```
Field[] fields1 = c.getFields();
Field[] fields2 = c.getDeclaredFields();
for(Field f1 : fields1) {
    bool = Arrays.asList(fields2).contains(f1); //fields2 contains f1?
    if (bool == true) {
        pvt_fields[i] = f1.getName(); //add to private variables
        i = i + 1;
    }
}
for (j = 0; j < i; j++) {
    System.out.println(pvt_fields[j]);
}
```

○
```
Field[] fields1 = c.getFields();
Field[] fields2 = c.getDeclaredFields();
for(Field f1 : fields1) {
    bool = Arrays.asList(fields2).contains(f1); //fields2 contains f1?
    if (bool == false) {
        pvt_fields[i] = f1.getName(); //add to private variables
        i = i + 1;
    }
}
for (j = 0; j < i; j++) {
    System.out.println(pvt_fields[j]);
}
```

√
```
Field[] fields1 = c.getFields();
Field[] fields2 = c.getDeclaredFields();
for(Field f2 : fields2) {
    bool = Arrays.asList(fields1).contains(f2); //fields1 contains f2?
    if (bool == false) {
```

```
            pvt_fields[i] = f2.getName(); //add to private variables
            i = i + 1;
        }
    }
    for (j = 0; j < i; j++) {
        System.out.println(pvt_fields[j]);
    }
```

Solution: The code in option 4 checks for each declared field, whether it is present in the list of fields returned by `getFields()` method. If any is not present, then it is a private field, and it is added to the list of private fields. The first two options return public instance variables, and the third option does not return anything.

BSCCS2005: Practice Assignment with Solutions
Week 6

1. Consider the code given below. [MCQ:2 points]

```
import java.util.*;
public class FClass{
    public static void main(String[] args) {
        ArrayList<String> empList = new ArrayList<String>();
        empList.add("raj");
        empList.add("akash");
        empList.add("biraj");
        empList.add("vinay");
        ------------------------------    //LINE 1
        while(iter.hasNext()) {
            System.out.print(iter.next() + " ");
        }
        System.out.println();
        while(iter.hasPrevious()) {
            System.out.print(iter.previous()  + " ");
        }
    }
}
```

Identify the appropriate option to fill in the blank at `LINE 1`, such that the output of the above code is

```
raj akash biraj vinay
vinay biraj akash raj
```

- ◯ `Iterator<String> iter = empList.iterator();`
- √ `ListIterator<String> iter = empList.listIterator();`
- ◯ `Iterator<String> iter = empList.listIterator();`
- ◯ `ListIterator<String> iter = empList.iterator();`

Solution: Since it requires the capabilities `hasPrevious()` and `previous()`, it need to use an iterator of object of type `ListIterator`, which can be achived using `listIterator()` function.

2. Consider the code given below. [MSQ:2 points]

```
import java.util.*;
public class Process{
    private int pid;
    public Process(int pid) {
        this.pid = pid;
    }
    public int getPID() {
        return pid;
    }
}
public class FClass {
    public static void main(String[] args){
        Queue<Process> pq = new LinkedList<Process>();
        for (int i = 0; i < 5; i++)
            pq.add(new Process(i + 1000));

        while(!pq.isEmpty()) {
            Process curPorc = _____________;    //LINE 1
            System.out.print(curPorc.getPID() + " -> ");
        }
    }
}
```

Identify the appropriate option(s) to fill in the blank at `LINE 1`, such that the output is:
`1000 -> 1001 -> 1002 -> 1003 -> 1004 ->`

- ◯ `pq.peek()`
- √ `pq.remove()`
- √ `pq.poll()`
- ◯ `pq.element()`

Solution: Since the `while` loop checks for if the queue is empty, within the loop the statement at `LINE` must return the element from the front of the queue and remove it from the queue. Thus, option-2 and option-3 are correct.

3. Consider the following code. [MCQ:2 points]

```
import java.util.*;
public class Shop{
  private String name;
  private int nsold; // number of items sold
  public Shop(String s, int ns){
    this.name = s;
    this.nsold = ns;
  }
  public String getName(){
    return name;
  }
  public int getItemSold(){
    return nsold;
  }
}

public class Test {
  public static void main(String[] args) {
    Shop s1 = new Shop("BigBazaar", 20);
    Shop s2 = new Shop("BigBazaar", 20);
    Shop s3 = new Shop("SV stores", 12);
    Shop s4 = new Shop("SunGeneral", 10);
    HashMap<String, Integer> m = new HashMap<String, Integer>();
    m.put(s1.getName(), m.getOrDefault(s1.getName(),0)+s1.getItemSold());
    m.put(s2.getName(), m.getOrDefault(s2.getName(),0)+s2.getItemSold());
    m.put(s3.getName(), m.getOrDefault(s3.getName(),0)+s3.getItemSold());
    m.put(s4.getName(), m.getOrDefault(s4.getName(),0)+s4.getItemSold());
    String shop = "";
    int sold = 0;
    -----------SEGMENT 1----------------
  }
}
```

At the end of execution, the variable `shop` should store the name of shop, which has sold maximum number of items, and the variable `sold` should store the total number of items sold by that shop. (In the case of the given code, `shop` should store BigBazaar and `sold` should store 40).

Identify the appropriate option to fill in the blank at `SEGMENT 1`.

◯ `for (HashMap.Entry<String, Integer> entry : m.entrySet()){`
`if(entry.getValue()> shop) {`

```
            shop = entry.getKey();
            sold = entry.getValue();
        }
    }
```

◯
```
for (HashMap.Entry<String, Integer> entry : m.entrySet()){
    if(entry.getKey()> sold) {
        shop = entry.getKey();
        sold = entry.getValue();
    }
}
```

◯
```
for (HashMap.Entry<Integer, Integer> entry : m.entrySet()){
    if(entry.getValue()> sold) {
        shop = entry.getKey();
        sold = entry.getValue();
    }
}
```

√
```
for (HashMap.Entry<String, Integer> entry : m.entrySet()){
    if(entry.getValue()> sold) {
        shop = entry.getKey();
        sold = entry.getValue();
    }
}
```

Solution: `HashMap m` maps the name of the shop to total number of items sold by that shop. `sold` value must be updated only if the entry.getValue() > sold.

4. Consider the following code. [MCQ:2 points]

```
import java.util.*;
public class Student{
  private String name;
  private int maths, physics, chemistry;
  Student(String s, int m, int p, int c){
    this.name = s;
    this.maths = m;
    this.physics = p;
    this.chemistry = c;
  }
  public String getName(){
    return name;
  }
  public int getMaths(){
    return maths;
  }
  public int getPhysics(){
    return physics;
  }
}
public class Test {
  public static void main(String[] args) {
    Student s1 = new Student("Ravi", 90, 55, 50);
    Student s2 = new Student("Ram" , 72, 80, 55);
    Student s3 = new Student("Ramu" , 50, 80, 55);
    ArrayList<Student> l1 = new<Student> ArrayList();
    ArrayList<Student> l2 = new<Student> ArrayList();
    l1.add(s1);
    l1.add(s2);
    l1.add(s3);
    for(Student s : l1){
      if(s.getMaths() > 80 && s.getPhysics() < 60){
        l2.add(s);
      }
    }
  }
}
```

Choose the correct option regarding the code.

- ◯ `l1` has 3 elements in it and `l2` has 3 elements in it.
- ◯ Compilation error because Student objects cannot be inserted in ArrayList

◯ Compilation error because > operator is not defined for Student type

√ `l1` has 3 elements in it and `l2` has 1 element in it

Solution: `ArrayList l1` has 3 elements. elements from `ArrayList l1` are added to `ArrayList l2` if the elements of `ArrayList l1` has maths marks > 80 and physics marks < 60.

5. Consider the following code. [MCQ:2points]

```
import java.util.*;
public class Test{
  public static void main(String args[]) {
    LinkedList<String> obj = new LinkedList<String>();
    obj.add("A");
    obj.add("C");
    obj.add(0, "D");
    obj.add("B");
    Collections.sort(obj);
    System.out.println(obj);
  }
}
```

What will the output be?

- √ [A, B, C, D]
- ◯ [D, A, C, B]
- ◯ [A, C, D, B]
- ◯ [A, D, B, C]

Solution: `Collections.sort(obj)` sorts the elements of `obj` in ascending order.

6. Consider the Java program given below and choose a possible outcome of executing it.

[MCQ : 2 points]

```
import java.util.*;
public class Example {
    public static void main(String[] args) {
        List<String> list1=new ArrayList<String>();
        list1.add("IITM");
        list1.add("Java");
        list1.add("Java");
        list1.add("Programming");
        Set<String> set1=new HashSet<String>(list1);//Line 1
        for (String string : set1) {
            System.out.println(string);
        }
    }
}
```

- ◯ Compile time error at Line 1
- √ This code generates output:

  ```
  Java
  Programming
  IITM
  ```

- ◯ This code generates output:

  ```
  Java
  Java
  Programming
  IITM
  ```

- ◯ This code generates output:

  ```
  null
  null
  null
  ```

Solution: Here, we pass an `ArrayList` object in the `HashSet` constructor in Line 1. Duplicate elements are skipped while adding elements to the set.

7. Consider the Java code given below and predict the output for Lines 1, 2, 3 and 4.

[MCQ : 2 points]

```
import java.util.*;
public class  Example{
    public static void main(String[] args){
        ArrayList<Integer> list=new ArrayList<Integer>();
        list.add(100);
        list.add(200);
        list.add(300);
        System.out.println(list.indexOf(100));//Line 1
        System.out.println(list.get(1));//Line 2
        HashSet<Integer> set=new HashSet<Integer>(list);
        System.out.println(set.indexOf(100));//Line 3
        System.out.println(set.get(2));//Line 4
    }
}
```

- ◯ Line 1 prints 0
  Line 2 prints 200
  Line 3 prints 0
  Line 4 prints 300
- ◯ Line 1 prints 1
  Line 2 prints 100
  Line 3 prints 1
  Line 4 prints 200
- ◯ Line 1 prints 0
  Line 2 prints 100
  Line 3 prints true
  Line 4 prints 300
- √ Compilation errors at Line 3 and 4.

Solution: A `List` interface is an ordered collection. All classes that implement the `List` interface will guarantee a sorted order while storing the elements. Hence, we can invoke `indexOf()` method and `get()` method on such classes.
A `Set` interface, on the other hand, is an unordered collection. The classes implementing the `Set` interface do not guarantee sorted order for the elements. Hence, it is meaningless to invoke `indexOf()` and `get()` methods on such classes. Thus, Lines 3 and 4 gives compilation errors.

8. Consider the Java program given below. [MCQ : 2 points]

```
import java.util.*;
public class  ArrayDequeExample{
    public static void main(String[] args){
        ArrayDeque<String> deque1=new ArrayDeque<String>();
        deque1.push("IIT");
        deque1.push("Madras");
        deque1.push("Java");
        deque1.push("Object");
        deque1.push("Oriented");
        deque1.push("Programming");
        deque1.push("Language");
        ArrayDeque<String> deque2=new ArrayDeque<String>(deque1);
        for (int i=0;i<6;i++) {
           deque1.pop();
        }
        for (int i=0;i<6;i++) {
           deque2.peek();
        }
    }
}
```

How many elements are present in `deque1` and `deque2` after executing this code?

- ◯ `deque1` has no elements.
  `deque2` has no elements.
- √ `deque1` has 1 element.
  `deque2` has 7 element.
- ◯ `deque1` has 1 element.
  `deque2` has 1 element.
- ◯ `deque1` has 7 elements.
  `deque2` has 7 elements.

Solution: 7 elements are present in both `deque1` and `deque2` before calling `pop()` and `peek()`.
`pop()` is called 6 times on `deque1`.
`peek()` is called 6 times on `deque2`.
`pop()` will return and remove the element from `deque1`, whereas `peek()` will return the value from `deque2` without removing it.

9. Consider the Java program given below, and choose correct the options.

[MCQ : 2 points]

```
import java.util.*;
public class MapEx{
    public static void main(String[] args) {
        Map<String,String> map1;
        map1=new HashMap<String,String>();       //Line 1
        map1.put("India","Delhi");
        map1.put("Srilanka","Colombo");
        map1.put("Australia","Sydney");
        System.out.println(map1);                //Line 2
        map1=new TreeMap<String,String>(map1);   //Line 3
        System.out.println(map1);                //Line 4
    }
}
```

- ◯ `Lines 2 and 4` always produce the same output.
- √ `Lines 2 and 4` may produce different output.
- ◯ Compilation error at `Line 1`.
- ◯ Compilation error at `Line 3`.

Solution: Applied indirection to the program.
`map1` reference variable instantiated with `HashMap` at `Line 1`.
`map1` reference variable instantiated with `TreeMap` at `Line 3`.
`map1` prints different output at `Line 2 and 4`.
`HashMap` can not store the values in sorted order.
`TreeMap` can store the values in sorted order.

10. Consider the code given below. Choose the correct option regarding the given code.

[MCQ:2 points]

```
import java.util.*;
public class Test{
    public static void main (String[] args){
        Map<String, Integer> map = new LinkedHashMap();
        String[] str = {"E","A","B","D","C"};
        Integer[] arr = {5,3,1,2,4};
        for(int i=0;i<str.length;i++){
           map.put(str[i],arr[i]);
        }
        Set s=map.entrySet();
        Iterator itr=s.iterator();
        while(itr.hasNext()){
           Map.Entry m = (Map.Entry)itr.next();
           if(m.getKey().equals("B")){
              m.setValue(2);
           }
           System.out.println(m.getKey()+" => "+m.getValue());
        }
    }
}
```

- ◯ This program generates output:
  A => 3
  B => 2
  C => 4
  D => 2
  E => 5
- √ This program generates output:
  E => 5
  A => 3
  B => 2
  D => 2
  C => 4
- ◯ This program generates output:
  B => 2
  D => 2
  A => 3
  C => 4
  E => 5
- ◯ This program generates compile time error since trying to assign same values to key `B` and `D`

Solution: The LinkedHashMap maintains the order in which key-value pairs are inserted.

BSCCS2005: Practice Assignment with Solutions
Week 7

1. Consider the code given below. [MCQ:2 points]

```
import java.util.logging.*;

public class FClass {
    public FClass() {
        Logger.getGlobal().config("cont called");
    }
    public void fun() {
        Logger.getGlobal().fine("fun() called");
    }
    public static void main(String[] args){
        FClass obj = new FClass();
        Logger.getGlobal().setLevel(Level.FINE);
        ConsoleHandler handler = new ConsoleHandler();
        handler.setLevel(Level.FINE);
        Logger.getGlobal().addHandler(handler);
        obj.fun();
        Logger.getGlobal().config("fun() return");
        Logger.getGlobal().log(Level.FINER, "end of main()");
    }
}
```

Identify the result when the code gets executed.

- ◯ It prints nothing
- ◯ <date time> FClass <init>
  CONFIG: cont called
  <date time> FClass fun
  FINE: fun() called
  <date time> FClass main
  CONFIG: fun() return
  <date time> FClass main
  FINER: end of main()
- √ <date time> FClass fun
  FINE: fun() called
  <date time> FClass main
  CONFIG: fun() return
- ◯ <date time> FClass fun
  FINE: fun() called
  <date time> FClass main
  CONFIG: fun() return
  <date time> FClass main
  FINER: end of main()

Solution: By default, three levels - `SEVERE`, `WARNING` and `INFO` - are logged and printed on the console. Thus, the string `"cont called"` is not logged.
After that, global logger is set log up to level `FINE`, and setting the level of `ConsoleHandler` object enables the logged data to be printed in console. Thus, the strings `FINE: "fun() called"` and `"CONFIG: fun() return"` would be logged as well as get printed on the console.
However, the string `"end of main()"` would not be logged, since level is set to `FINER` which is at lower than level `FINE`.

2. Consider the code given below. [MCQ:2 points]

```
import java.util.logging.*;

public class SomeClass {
    private final static Logger logbook =
                        Logger.getLogger(SomeClass.class.getName());
    public void logIt(){
        logbook.info("logIt() called");
        logbook.setLevel(Level.INFO);
    }
}

public class FClass {
    private final static Logger logbook =
                        Logger.getLogger(FClass.class.getName());
    public static void main(String[] args){
        logbook.info("main() started");
        logbook.setLevel(Level.WARNING);
        SomeClass obj = new SomeClass();
        obj.logIt();
        logbook.info("main() ends");
    }
}
```

Identify the result when the code gets executed.

√ <date time> FClass main
INFO: main() started
<date time> SomeClass logIt
INFO: logIt() called

◯ <date time> FClass main
INFO: main() started
<date time> SomeClass logIt
INFO: logIt() called
<date time> FClass main
INFO: main() ends

◯ <date time> FClass main
INFO: main() started

◯ <date time> FClass main
INFO: main() started
<date time> PM FClass main
INFO: main() ends

Solution: The program logger two different logger for two different classes.
In class `FClass`, logger name is `FClass`. Since, by default, `INFO` level messages is logged, the message `"main() started"` is logged and printed. However, the next statement set the log level to `WARNING`, which is higher level than `INFO`. Thus, next message `main() ends"` would not be printed.
In class `SomeClass`, logger name is `SomeClass`. Since, by default, `INFO` level messages is logged, the message `"logIt() started"` is logged and printed.

3. Consider the Java code given below. [MCQ:2 points]

```
public class MainClass{
    public static void main(String[] args){
        int a = 10;
        int b = 0;
        try{
            assert b != 0;
            int c = a / b;
        }
        catch(ArithmeticException e){
            System.out.println("in catch block");
        }
        finally{
            System.out.println("in finally block");
        }
        System.out.println("end of main()");
    }
}
```

Choose the most appropriate option regarding the code when executed as:
`java -ea MainClass`

- ◯ It only provides a runtime exception: `java.lang.AssertionError`
- √ It prints: `in finally block`,
  then provides a runtime exception: `java.lang.AssertionError`
- ◯ It prints:
  `in finally block`
  `end of main()`,
  then provides a runtime exception: `java.lang.AssertionError`
- ◯ It prints:
  `in catch block`
  `in finally block`
  `end of main()`

Solution: The `assert` statement in `try` block throws an `AssertionError`. However, there is no `catch` block to handle the exception. So it causes abrupt termination of the program. But, before termination it executes the `finally` block, then terminates after providing runtime exception information.

4. Consider the Java code given below. [MCQ:2 points]

```
public class DOBRegistration{
    private int day, month, year;
    public DOBRegistration(int day, int month, int year){
        assert 0 < day && day <= 31: "day :" + day;      //assert-1
        this.day = day;
        assert 0 < month && month <= 12: "day :" + day;  //assert-2
        this.month = month;
        this.year = year;
    }
}
public class JobApplication{
    private int age, exp;
    public JobApplication(int age, int exp){
        assert age >= 18: "invalid age for job";         //assert-3
        this.age = age;
        assert exp >= 3: "invalid experience for job";   //assert-4
        this.exp = exp;
    }
}
public class FClass{
    public static void main(String[] args){
        DOBRegistration dr = new DOBRegistration(2, 23, 1879);
        JobApplication ja = new JobApplication(20, 1);
    }
}
```

Identify the `assert` statement that throws the `AssertionError` when the class is executed as:
`java -ea:JobApplication FClass`

- ◯ `assert-1`
- ◯ `assert-2`
- ◯ `assert-3`
- √ `assert-4`

Solution: Since assertions are enabled for class `JobApplication`, assert statement `assert-4` throws `AsserionError`.

5. Consider the following Java code and choose the correct option. [MCQ : 2 points]

```
public class Example{
    public static void main(String[] args) {
        int a[]=new int[10];
        try{
            a[10]=50;
            System.out.println("value in the array is "+a[10]);
        }
        catch (ArrayIndexOutOfBoundsException ae){
            System.out.println("Please check array index");
        }
        finally{
            System.out.println("end of main()");
        }
    }
}
```

- ◯ This program generates output:
  `value in the array is 50`
  `end of main()`
- ◯ This program generates output:
  `Please check array index`
- √ This program generates output:
  `Please check array index`
  `end of main()`
- ◯ This program generates output:
  `end of main()`

Solution: In above program, the statement a[10] = 50; throws ArrayIndexOutOf-BoundException. The corresponding catch block statements will be executed. After catch block, finally block statements will be executed.

6. Consider the following Java code and choose the correct option. [MSQ : 2 points]

```
public class Example{
    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);
        int a,b,c;
        a=scanner.nextInt();
        b=scanner.nextInt();
        String name = "IIT madras java";
        int index=scanner.nextInt();
        try{
            c = a/b;
            System.out.println("Quotient is "+c);
            System.out.println(name.charAt(index));
        }
        catch(ArithmeticException | ArrayIndexOutOfBoundsException e){
            System.out.println(e);
        }
    }
}
```

If `b` is 0 and `index` is greater than the length of `name`, then what will the outcome be?

- ◯ The program terminates due to unhandled exception(s).
- √ It prints a message on `ArithmeticException`.
- ◯ It prints a message on whichever exception occurs first.
- ◯ It prints messages on both the exceptions.

Solution:

```
catch(ArithmeticException | ArrayIndexOutOfBoundsException e){
System.out.println(e);
}
```

Above is an alternative for the multi catch block. Both the exceptions can be handled. But only one exception can be handled at a time.

7. Consider the following Java code and choose the correct option.

[MCQ : 2 points]

```
public class A{
    public void show() throws Exception{
        System.out.println("Super class show with Exception");
    }
}
public class B extends A{
    public void show() throws RuntimeException{
        System.out.println("Sub class show with RuntimeException");
    }
}
public class Example {
    public static void main(String[] args) {
        B ob=new B();
        ob.show();
    }
}
```

- ◯ Compilation error because we cannot override the `show()` method in subclass B, as the `show()` method is throwing the more general exception, namely, `Exception` in super class A.
- ◯ Compilation error because we cannot override the `show()` method in subclass B, as the `show()` method is throwing the `RuntimeException`, which is not compatible with `Exception`.
- √ This code generates the output:
  `Sub class show with RuntimeException`
- ◯ This code generates the output:
  `Super class show with Exception`
  `Sub class show with RuntimeException`

Solution: You can override `show()` in subclass B by throwing RuntimeException. ob.show() will call `show()` from the subclass B.

8. Consider the following Java code and choose the correct option.

[MCQ : 2 points]

```
public class  Example{
    int show(){
        try{
            return 10;
        }
        catch (Exception e){
            return 20;
        }
        finally {
            return 100;
        }
    }
    public static void main(String[] args) {
        Example object=new Example();
        System.out.println(object.show());
    }
}
```

- √ This code generates the output:
  100
- ◯ This code generates the output:
  20
- ◯ This code generates the output:
  0
- ◯ Compilation failed.

Solution: Here, the try, catch and finally blocks are returning values to the method. In such a scenario, whatever value is returned by the finally block will be considered as the final return value.

9. Consider the following Java code and choose the correct option.

[MCQ : 2 points]

```
public class  Example{
    public void show() throws Exception{
        System.out.println("show() with Exception");
    }
    public void show(int a) throws RuntimeException{
        try {
            show();
        }
        catch (Exception e) {
        }
        System.out.println("show() overloaded with RuntimeException");
    }
    public void show(String a){
        show(10);
        System.out.println("show() overloaded with no Exceptions");
    }
    public static void main(String[] args){
        try{
            new Example().show("IITM Java");
        }
        catch (Exception e){
            System.out.println(e);
        }
    }
}
```

- ◯ Compilation error
- ◯ The program terminates abnormally because of ambiguity in the overloading of the show() method.
- √ This program generates the output:

```
show() with Exception
show() overloaded with RuntimeException
show() overloaded with no Exceptions
```

- ◯ This program generates the output:

```
show() overloaded with no Exceptions
show() overloaded with RuntimeException
show() with Exception
```

Solution: Option 3 is correct
We can overload a method which throws `Exception`.
The overloading method may not throw any exception, or it may throw the same or a different exception as the original method.

10. Consider the following Java code and choose the correct way of using the class Example from any outside package.

[MSQ : 2 points]

```
package iitm.java.program;
public class  Example{
    public void show(){
        System.out.println("This is show");
    }
}
```

√
```
import iitm.java.program.Example;
public class UseExample {
    public static void main(String[] args){
        new Example().show();
    }
}
```

◯
```
public class UseExample {
    public static void main(String[] args){
        new Example().show();
    }
}
```

√
```
public class UseExample {
    public static void main(String[] args){
        new iitm.java.program.Example().show();
    }
}
```

◯
```
import iit.*;
public class UseExample {
    public static void main(String[] args){
        new Example().show();
    }
}
```

Solution: Options 1 and 3 are correct.
Option 1 imports the package using import keyword.
Option 3 imports the package using the fully qualified name.

11. Consider the following two source code files located in two different packages as shown.

[MSQ : 2 points]

//Adder.java

```
package iitm;
public class Adder {
    public int add(int n1, int n2, int n3) {
        return n1+n2+n3;
    }
    protected int add(int n1,int n2) {
        return n1+n2;
    }
}
```

//Test1.java

```
package test;
import iitm.*;

class Calculator extends Adder{
    public void calculate() {
        System.out.println(this.add(7,8,9)); // LINE 1
        System.out.println(this.add(9,10));  // LINE 2
    }
}
public class Test1{
    public static void main(String args[]) {
        Adder a1 = new Adder();

        System.out.println(a1.add(4,5));   // LINE 3
        System.out.println(a1.add(1,2,3)); // LINE 4

        new Calculator().calculate();
    }
}
```

Choose the correct option regarding these two .java files.

- ◯ LINE 1 will lead to compilation error.
- ◯ LINE 2 will lead to compilation error.
- √ LINE 3 will lead to compilation error.
- ◯ LINE 4 will lead to compilation error.

Solution: `protected` members of a class can only be accessed in subclasses within the package as well as outside the package. Members of a class where access specifier is no mentioned explictly is treated as *package private* types and can only be accessed within that specific package. `public` members are accessible through out all packages and all classes
Therefore the `Adder` object `a1` inside the `test1` class's `main` method can only access public members of `Adder` class.

BSCCS2005: Practice Assignment with Solutions
Week 8

1. Consider the code given below. [MCQ:2 points]

```
public class ComplexNum implements Cloneable{
    private double r, i;
    public ComplexNum(double r, double i) {
        this.r = r;
        this.i = i;
    }
    public void setRe(double r) {
        this.r = r;
    }
    public void setIm(double i) {
        this.i = i;
    }
    public String toString() {
        return "(" + r + " + " + i + "i)";
    }
    public Object clone() throws CloneNotSupportedException{
        return super.clone();
    }
}

public class FClass{
    public static void main(String[] args) {
        try {
            ComplexNum c1 = new ComplexNum(10.0, 20.0);
            ComplexNum c2 = c1;
            ComplexNum c3 = (ComplexNum)c1.clone();
            c1.setRe(100.0);
            c1.setIm(200.0);
            System.out.println(c1 + " , " + c2 + ", " + c3);
        }
        catch(CloneNotSupportedException e) {
            System.out.println("clone() not supported");
        }
    }
}
```

What will the output be?

- ◯ `(100.0 + 200.0i) , (100.0 + 200.0i), (100.0 + 200.0i)`
- √ `(100.0 + 200.0i) , (100.0 + 200.0i), (10.0 + 20.0i)`
- ◯ `(100.0 + 200.0i) , (10.0 + 20.0i), (10.0 + 20.0i)`
- ◯ `clone() not supported`

Solution: Since, `c1` and `c2` refers to the same object, any change to `c1` would be reflected on `c2`. However, `c3` creates a separate copy of the `c1` object. Thus, the changes in `c1` are not reflected on `c3`.

2. Consider the code given below. [MCQ:2 points]

```
public class Product{
    private String prodname;
    private double prodprice;
    public Product(String prodname, double prodprice) {
        this.prodname = prodname;
        this.prodprice = prodprice;
    }
    public void updateProduct(String prodname, double prodprice) {
        this.prodname = prodname;
        this.prodprice = prodprice;
    }
    public String toString() {
        return prodname + " : " + prodprice;
    }
}

public class Order implements Cloneable{
    private int orderid;
    private Product prod;
    public Order(int orderid, Product prod) {
        this.orderid = orderid;
        this.prod = prod;
    }
    public Order clone() throws CloneNotSupportedException{
        return (Order)super.clone();
    }
    public void updateOrder(int orderid, String prodname, double prodprice) {
        this.orderid = orderid;
        prod.updateProduct(prodname, prodprice);
    }
    public String toString() {
        return orderid + " : " + prod;
    }
}

public class FClass{
    public static void main(String[] args) {
        try {
            Order od1 = new Order(1001, new Product("Pen", 15.0));
            Order od2 = od1.clone();
            od1.updateOrder(1010, "Pencil", 20.0);
            System.out.print(od1 + ", " + od2);
```

```
        }
        catch(CloneNotSupportedException e) {
            System.out.println("clone() not supported");
        }
    }
}
```

What will the output be?

- √ `1010 : Pencil : 20.0, 1001 : Pencil : 20.0`
- ◯ `1010 : Pencil : 20.0, 1001 : Pen : 15.0`
- ◯ `1010 : Pencil : 20.0, 1010 : Pencil : 20.0`
- ◯ `clone() not supported`

Solution: Since the instance variable `prod` in `Order` class is a reference type, The bitwise copy made by `od2.clone()` copies the reference. Thus, although `od1` and `od2` refers to different object, both the object hold a reference `prod` referring to the same memory holding `Product` object. Thus, update on `od1.orderid` would not be reflected on `o2.orderid`; however, update on `od1.prod` would be reflected on `o2.prod`.

3. Consider the code given below. [MCQ:2 points]

```
public class Address implements Cloneable{
    private int houseno;
    private String city;
    public Address(int houseno, String city) {
        this.houseno = houseno;
        this.city = city;
    }
    public void updateAddress(int houseno, String city) {
        this.houseno = houseno;
        this.city = city;
    }
    public String toString() {
        return houseno + " : " + city;
    }
    public Address clone() throws CloneNotSupportedException{
        return (Address)super.clone();
    }
}

public class Person implements Cloneable{
    private String name;
    private Address addr;
    public Person(String name, Address addr){
        this.name = name;
        this.addr = addr;
    }
    public Person clone() throws CloneNotSupportedException{
        Person newPer = (Person)super.clone();
        newPer.addr = addr.clone();
        return newPer;
    }
    public void updatePerson(String name, int houseno, String city) {
        this.name = name;
        addr.updateAddress(houseno, city);
    }
    public String toString() {
        return name + " : " + addr;
    }
}

public class FClass{
    public static void main(String[] args) {
```

```
        try {
            Person per1 = new Person("binit", new Address(100, "Delhi"));
            Person per2 = per1.clone();
            per1.updatePerson("rajiv", 200, "Kolkata");
            System.out.print(per1 + ", " + per2);
        }
        catch(CloneNotSupportedException e) {
            System.out.println("clone() not supported");
        }
    }
}
```

What will the output be?

- ◯ `rajiv : 200 : Kolkata, binit : 200 : Kolkata`
- ◯ `rajiv : 200 : Kolkata, rajiv : 200 : Kolkata`
- √ `rajiv : 200 : Kolkata, binit : 100 : Delhi`
- ◯ `clone() not supported`

Solution: The `clone()` method of `Person` performs a deep copy, i.e. it clones the `Address` type instance variable `addr` also. Thus, any update on `per1` would not be reflected on `per2`.

4. Consider the code given below. [MSQ:2 points]

```
public class FClass{
    public static void main(String[] args) {
        _________________________    //LINE 1
        for(var i : iArr)
            System.out.print(i + " ");
    }
}
```

At `LINE 1`, identify the appropriate definition of the array `iArr`, such that the output is:

`10 20 30 40 50`

- √ `int[] iArr = new int[] {10, 20, 30, 40, 50};`
- √ `var iArr = new int[] {10, 20, 30, 40, 50};`
- ◯ `var[] iArr = new int[] {10, 20, 30, 40, 50};`
- ◯ `var iArr = {10, 20, 30, 40, 50};`

Solution: Among the given options, option-3 and option-4 are wrong syntax using `var`.

5. Consider the code given below. [MSQ:2 points]

```
import java.util.*;
public class Person{
    private String name;
    private int age;
    public Person(String n, int a) {
        name = n;
        age = a;
    }
    public int getAge(){
        return age;
    }
    public void print() {
        System.out.println(name + " : " + age);
    }
}

public class FClass{
    public static void main(String[] args) {
        var list = new ArrayList<Person>();
        list.add(new Person("Robin", 33));
        list.add(new Person("Indra", 76));
        list.add(new Person("Smita", 35));
        list.add(new Person("Rikki", 26));
        Collections.sort(list, ________________________);
        for(var l: list)
            l.print();
    }
}
```

Identify the appropriate option(s) to fill in the blank at `LINE 1`, such that the output is:

```
Indra : 76
Smita : 35
Robin : 33
Rikki : 26
```

- ◯ `(a, b) -> a.getAge() - b.getAge()`
- √ `(a, b) -> b.getAge() - a.getAge()`
- ◯ `(Person a, Person b) -> a.getAge() - b.getAge()`
- √ `(a, b) -> { return b.getAge() - a.getAge(); }`

Solution: Option 1 – lamda expression `(a, b) -> a.getAge() - b.getAge()` and option 3 – lamda expression `(Person a, Person b) -> a.getAge() - b.getAge()` are syntactically correct. However, they sort the `Person` objects in ascending order of `age`.
Option 2 – lamda expression `(a, b) -> b.getAge() - a.getAge()` and option 4 – lamda expression `(a, b) -> { return b.getAge() - a.getAge(); }` are syntactically correct and they sort the `Person` objects in descending order of `age`.

6. Consider the code given below. [MCQ:2 points]

```
import java.util.*;
public interface Operatable<T extends Number>{
    public T operate(T a);
}

public class FClass{
    ------------------------------------------------ {    //LINE 1
        for(int i = 0; i < x.length; i++)
            x[i] = ob.operate(x[i]);
    }
    public static void main(String[] args) {
        Integer[] iArr = new Integer[]{1, 2, 3, 4, 5};
        map(iArr, i -> i * i);
        for(int i: iArr)
            System.out.print(i + " ");
    }
}
```

Identify the appropriate option(s) to fill in the blank at `LINE 1`, such that the output is:

```
1 4 9 16 25
```

- ◯ `public static <T> void map(T[] x, Operatable<T> ob)`
- √ `public static <T extends Number> void map(T[] x, Operatable<T> ob)`
- ◯ `public static void map(Integer[] x, Operatable<?> ob)`
- ◯ `public static <T extends Number> void map(Operatable<T> ob, T x[])`

Solution: Since `operate` method in `Operatable` applicable to any subtype of `Number`, the `for_each` function must define a generic parameter type that extends `Number`. As per the call made from `main` – `map(i_Arr, i -> i * i)`, the first argument of `for_each` must be a generic array, and second argument must be an object of `Operatable` type.

7. The `merge` method of `Map` has three arguments - key, value and reference to a function accepting two arguments - and merges the old value with the new value for a given key. Consider the code given below. [MSQ:2 points]

```
import java.util.*;
public class FClass{
    public static void main(String[] args){
        Map<String, Double> prices = new LinkedHashMap<String, Double>();
        prices.put("Pen", 30.0);
        prices.put("Pencil", 10.0);
        prices.put("Notebook", 40.0);
        prices.put("Paper", 5.0);

        for(Map.Entry<String, Double> e : prices.entrySet())
            prices.merge(__________________________________);

        System.out.println(prices);
    }
}
```

Identify the appropriate option to fill in the blank at `LINE 1`, such that the output is:

```
{Pen=33.0, Pencil=11.0, Notebook=44.0, Paper=5.5}
```

- ✓ `e.getKey(), 0.1, (x, y) -> x + x * y`
- ◯ `e.getKey(), 0.1, (y, x) -> x + x * y`
- ◯ `e.getKey(), 0.1, (x, y) -> y + x * y`
- ✓ `e.getKey(), 0.1, (y, x) -> y + x * y`

Solution: The computation of new value (`new_value`) of for each (key, value) pair would be calculated as:
`new_value = (old_value, 0.1) -> old_value + old_value * 0.1`

8. Which of the following statements can find out the number of integers between 0 to 50 that are divisible by 3? [MCQ:2 points]

○
```
long c = Stream.iterate(0, n -> n + 3)
               .limit(50).count();
```

√
```
long c = Stream.iterate(0, n -> n + 3)
               .takeWhile(n -> n <= 50).count();
```

√
```
long c = Stream.iterate(0, n -> n <= 50, n -> n + 1)
               .filter(n -> n % 3 == 0).count();
```

○
```
long c = Stream.iterate(0, n -> n + 3)
               .dropWhile(n -> n < 50).count();
```

Solution: The question asked is to determine the number of elements that are in between 0 and 50 and divisible by 3.
Option-1 – generate total 50 elements from 0 which are divisible by 3.
Option-2 and -3 options are correct.
Option-4 – starts generating the numbers divisible by 3 from 50.

9. Consider the Java code given below that adds a list of students and their marks to a TreeMap.

[MCQ:2pts]

```
import java.util.stream.Stream;
import java.util.*;

public class TreeStream{
    public static void main(String []args){
        Map<Integer,String> student_map = new TreeMap<Integer,String>();

        student_map.put(80,"Arya");
        student_map.put(62,"Diya");
        student_map.put(71,"Fiona");
        student_map.put(79,"Mason");
        student_map.put(90,"Maria");

        Stream<Map.Entry<Integer,String>> scores
                            = student_map.entrySet().stream().limit(3);
        System.out.println(Arrays.toString(scores.toArray()));
    }
}
```

What will the output be?

- √ [62=Diya, 71=Fiona, 79=Mason]
- ◯ [80=Arya, 62=Diya, 71=Fiona]
- ◯ [90=Maria, 80=Arya, 79=Mason]
- ◯ [71=Fiona, 79=Mason, 90=Maria]

Solution: The list is sorted inside the TreeMap in the increasing order of the keys, which are the scores here. `limit(3)` extracts only the first three, which will be the list of students with the least scores.

10. From among the options, choose the code segment that gives the same output as is given by the Java code inside the `CODE BLOCK`.

[MSQ:2pts]

```
import java.util.stream.Stream;
import java.util.*;
import java.util.stream.Collectors;

public class FClass {
    public static void main(String[] args) {
      //CODE BLOCK begins here
      Stream<Integer> integers = Stream.iterate(0, i -> i < 50, i -> i+1);
      integers.map(i -> i % 7 == 0).forEach(System.out::println);
      //CODE BLOCK ends here
    }
}
```

◯
```
Stream<Integer> integers = Stream.iterate(0, i -> i < 50, i -> i+1);
integers = integers.filter(i -> i % 7 == 0);
integers.forEach(System.out::println);
```

√
```
Stream.iterate(0, i -> i < 50, i -> i+1)
              .map(i -> i % 7 == 0)
              .forEach(System.out::println);
```

◯
```
Stream<Integer> integers
              = Stream.iterate(0, i -> i < 50, i -> i+1)
              .map(i -> i % 7 == 0)
              .forEach(System.out::println);
```

√
```
Stream<Integer> integers = Stream.iterate(0, i -> i < 50, i -> i+1);
List<Boolean> newList = integers.map(i -> i % 7 == 0)
                                .collect(Collectors.toList());
for (int i = 0; i < newList.size();i++){
    System.out.println(newList.get(i));
}
```

Solution: The `CODE BLOCK` maps each number to whether it is divisible by 7 or not. It prints the boolean value based on the result for each element. In Option 1, it is filtering out the ones that are divisible by 7, and displays the actual values that are divisible by 7.
Option 2 also prints the boolean value based on whether each element is divisible by 7 or not.
Option 3 will give an error because forEach(System.out::println) returns void, and

hence it cannot be assigned to a Stream object. Option 4 assigns the boolean values to a List using method `collect(Collectors.toList())`, and prints the values by iterating through the list.

BSCCS2005: Practice Assignment with Solutions
Week 9

1. Consider the following Java code and choose the correct option.

```
import java.util.*;
public class Test{
   public static void main(String[] args){
      String[] arr=new String[10];
      arr[0]="Sun";
      arr[1]="Moon";
      Optional<String> op=Optional.ofNullable(arr[2]);
      op.ifPresent(n->System.out.println(n.toUpperCase()));
   }
}
```

- ◯ This program generates `ArrayIndexOutOfBoundsException`.
- ◯ This program generates `NullPointerException`.
- ◯ This program generates output:
  null
- √ This program does not generate any output.

Solution: `ifPresent(Consumer<? super T> consumer)` method invoke the specified `consumer` with value if the value is present in the `Optional`, otherwise does nothing.

2. Consider the code given below.

```
import java.util.*;
import java.util.stream.Stream;
public class OptionalExample{
    public static void main(String[] args){
        Optional<Double> maxvalue =
            Stream.generate(Math::random)
            .limit(100)
            .max(Double::compareTo);
        var list = new ArrayList<Double>();
        maxvalue.ifPresentOrElse(
                v -> System.out.println("max value found"),
                () -> System.out.println("No max")
            );
    }
}
```

Choose the correct option regarding the code.

- ◯ Compilation failed.
- √ This program generates the output.
  `max value found`
- ◯ This program generates output:
  `No max`
- ◯ This program generates no output.

Solution: Stream generates the 100 random values between 0 and 1, max() will return the maximum value among all random values. `ifPresentOrElse()` will check whether maxrand contain the value or not, if present it will print `max value found` else will print `No max`.

3. Consider the following Java code and choose the correct option.

```
import java.util.*;
import java.util.stream.*;
public class Test{
   public static void main(String[] args){
      var i = 10;
      var list = new ArrayList<Integer>();
      while(i>1){
         list.add(i);
         i = i-1;
      }
      Stream<Integer> stream=list.stream();
      Integer[] num=stream.filter(n->n<5).map(n->n*n).toArray(Integer[]::new);
      for(var x=0;x<4;x++){
         System.out.println(num[x]);
      }
   }
}
```

- ◯ This program generates compile time error because stream can not be converted into array.
- ◯ This program generates output:
  2
  3
  4
- √ This program prints 16, 9 and 4 followed by `ArrayIndexOutOfBoundsException`.
- ◯ This program generates output:
  4
  9
  16

4. Consider the following Java code and choose the correct option.

```
import java.util.*;
import java.util.stream.*;
public class Employee{
   int id;
   String name;
   int service;
   Employee(int id, String name, int service){
        this.id = id;
        this.name = name;
        this.service = service;
   }
   public int getId(){
        return id;
   }
   public String getName(){
        return name;
   }
   public int getService(){
        return service;
   }
   public String toString(){
        return "Employee{" + "id=" + id + ", Name=" + name + ", Service=" +
                service + '}';
   }
}
public class FClass{
    public static void main(String[] args){
        var employee=new ArrayList<Employee>();
        employee.add(new Employee(1,"Mercury",10));
        employee.add(new Employee(2,"Venus",5));
        employee.add(new Employee(3,"Earth",3));
        employee.add(new Employee(6,"Saturn",2));
        employee.add(new Employee(7,"Uranus",10));
        employee.add(new Employee(8,"Neptune",10));
        Map<Integer, List<Employee>> map = employee.stream().
                collect(Collectors.groupingBy(i->i.getService()));
        System.out.println(map.get(10).get(1));
    }
}
```

◯ This program generates compile time error because stream cannot be converted to map.

√ This program generates output:
Employee{Id=7, Name=Uranus, Service=10}

◯ This program generates `KeyNotFoundException`.

◯ This program generates output:
Employee{id=1, Name=Mercury, Service=10}

Solution: `groupingBy(Function<? super T,? extends K> classifier)` method returns a `Collector` implementing a "group by" operation on input elements of type T. It groups elements according to a classification function, and returns the results in a Map.

5. Consider the following Java code and choose the correct option.

```
import java.util.*;
import java.util.stream.*;
public class Planet{
    private String name;
    private int temp;
    public Planet(String name, int temp) {
        this.name = name;
        this.temp = temp;
    }
    public String getName() {
        return name;
    }
    public int getTemp() {
        return temp;
    }
}
public class FClass{
    public static void main(String[] args){
        List<Planet> planet=new ArrayList();
        planet.add(new Planet("Mercury",480));
        planet.add(new Planet("Venus",430));
        planet.add(new Planet("Earth",30));
        planet.add(new Planet("Mars",-25));

        Set<String> set = planet.stream().
                filter(n->n.getTemp()>450).
                map(x->x.getName()).collect(Collectors.toSet());

        Map<Integer, String> map = planet.stream().
                filter(x->x.getTemp()<0).
                collect(Collectors.toMap(a->a.getTemp(), b->b.getName()));

        set.forEach(System.out::println);
        for (Map.Entry entry : map.entrySet()){
            System.out.println("key: " + entry.getKey() +
            "; value: " + entry.getValue());
        }
    }
}
```

◯ This program generates output:
null

Mercury
key: -25; value: Mars

√ This program generates output:
Mercury
key: -25; value: Mars

◯ This program generates output:
Mercury
key: 480; value: Mercury
key: 430; value: Venus
key: 30; value: Earth
key: -25; value: Mars

◯ This program generates output:
Jupiter
Mercury
key: -25; value: Mars

6. Consider the code given below.
Assume that there is no file named "`E:\\Files\\earth.txt`".

```
import java.io.*;
public class Example {
    public static void main(String[] args) {
        try {
            var in=new FileInputStream("E:\\Files\\earth.txt");
            var din=new DataInputStream(in);
            System.out.println("Data from file:");
            System.out.println(din.readLine());
        }
        catch (FileNotFoundException e) {
            System.out.println("File does not exist.");
        }
        catch (IOException e) {
            System.out.println("Error in writing a file.");
        }
        finally {
            System.out.println("Program execution finished.");
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ Compilation error.
- ◯ This program terminates abnormally due to unhandled exception(s).
- ◯ This program generates output:
  `Error in writing a file.`
  `Program execution finished.`
- √ This program generates output:
  `File does not exist.`
  `Program execution finished.`

Solution: While writing data into the file, if there is no file exist, program throws FileNotFoundException.
catch block with FileNotFoundException is executed and finally block also executed in above program.

7. Consider the code given below.
Assume that the file "E:\\Files\\books.txt" contains the following text in it.
Hi I am already here.

```
import java.io.*;
public class Example {
    public static void main(String[] args) {
        try {
            var out=new FileOutputStream("E:\\Files\\books.txt",false);
            var dout=new DataOutputStream(out);
            String data="Hello I have added  to you.";
            dout.writeBytes(data);
            System.out.println("Data written to file successfully");
        }
        catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            System.out.println("Program execution finished.");
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ Compilation error.
- √ After program execution "E:\\Files\\books.txt" contains the following line of text in it.
  Hello I have added to you.
- ◯ After program execution "E:\\Files\\books.txt" contains the following line of text in it.
  Hi I am already here.
- ◯ After program execution "E:\\Files\\books.txt" contains the following line of text in it.
  Hi I am already here.Hello I have added to you.

Solution:

```
var out=new FileOutputStream("E:\\Files\\books.txt",false);
```

In the above statement, false indicates that you cannot append new data to existing data in a file.
only new data added to file by erasing existing data in a file.

8. Consider the code given below.
Assume that the file "`E:\\Files\\library.txt`" contains the following lines of text in it.

```
A library is a collection of books.
Library provides hard copies of documents.
Library provides digital access to materials.
```

```
import java.io.*;
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        try {
            var in=new FileInputStream("E:\\Files\\library.txt");
            var scanner=new Scanner(in);
            System.out.println("Data from file:");
            System.out.println(scanner.nextLine());
            System.out.println(scanner.next());
            System.out.println(scanner.nextLine());
        }
        catch (FileNotFoundException e) {
            System.out.println("File does not exist.");
        }
        catch (IOException e) {
            System.out.println("Error in writing a file.");
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ Compilation error.
- ◯ Program terminates abnormally due to unhandled exception(s).
- √ This program generates the output:

```
Data from file:
A library is a collection of books.
Library
 provides hard copies of documents.
```

- ◯ This program generates the output:

```
Data from file:
A library is a collection of books.
A
Library provides digital access to materials.
```

Solution: Assume books.txt contains the following lines of text in it.

```
A library is a collection of books.
Library provides hard copies of documents.
Library provides digital access to materials.
```

System.out.println(scanner.nextLine());
The above statement will read the first line.
System.out.println(scanner.next());
The above statement will read only one word from the second line.
System.out.println(scanner.nextLine());
The above statement will read reaming portion of the second line.

9. Consider the following Java code and choose the correct option.

```
import java.io.*;
public class X implements Serializable{
    String str="Moon";
}
public class Y extends X{
    transient String str2="Sun";
}
public class Test{
    public static void main(String[] args) throws Exception{
        ObjectOutputStream os = new ObjectOutputStream(new FileOutputStream
                                ("File.ser"));
        os.writeObject(new Y());
        FileInputStream fis=new FileInputStream("File.ser");
        ObjectInputStream ois=new ObjectInputStream(fis);
        Y e=(Y)ois.readObject();
        System.out.print(e.str+"\n"+e.str2);
    }
}
```

- ◯ This program throws compile time error because `Y` does not implement `Serializable`.
- ◯ This program compiles successfully but throws `NotSerializableException` at runtime.
- ◯ This program generates output:
  Moon
  Sun
- √ This program generates output:
  Moon
  null

Solution: The serializable nature is inheritable. Thus, eventhough a child class does not implements `Serializable`, the child class objects by default gets serialized if the parent class implements `Serializable`.

10. Consider the following Java code and choose the correct option.

```
import java.io.*;
public class Mail implements Serializable{
    String user="Moon@mail.sun";
    transient int pass=1234;
    private void writeObject(ObjectOutputStream oos) throws Exception{
        oos.defaultWriteObject();
        int encrypt=(pass*100)+10;
        oos.writeObject(encrypt);
    }
    private void readObject(ObjectInputStream ois) throws Exception{
        ois.defaultReadObject();
        int decrypt=(int)ois.readObject();
        pass=(decrypt/100);
    }
}
public class Test{
    public static void main(String[] args) throws Exception{
        FileOutputStream fos=new FileOutputStream("File.ser");
        ObjectOutputStream oos=new ObjectOutputStream(fos);
        oos.writeObject(new Mail());
        FileInputStream fis=new FileInputStream("File.ser");
        ObjectInputStream ois=new ObjectInputStream(fis);
        Mail m=(Mail)ois.readObject();
        System.out.println(m.user+"\n"+m.pass);
    }
}
```

- √ This program generates output:
  Moon@mail.sun
  1234
- ◯ This program generates output:
  Moon@mail.sun
  1244
- ◯ This program generates output:
  Moon@mail.sun
  0
- ◯ This program generates output:
  Moon@mail.sun
  null

BSCCS2005: Practice with Solutions
Week 10

Two passengers P_1 and P_2 are trying to book sleeper class tickets on the same train. Suppose there is exactly one sleeper ticket left on that train and that there is no provision to book RAC and/or waiting list tickets. Consider the Java code modeling the given scenario, and answer questions 1 and 2.

```java
class Passenger implements Runnable{
    private String name;
    private int coach_no, seat_no;
    public boolean booked = false;
    public Passenger(String pass_name) {
        name = pass_name;
    }
    public void run() {
        coach_no = 10;
        seat_no = 52;
        booked = true;
    }
    public String getSeatAndPNR() {
        return (coach_no+" "+seat_no);
    }
    public String getName() {
        return name;
    }
}
public class RailwayBooking {
        public static void main(String args[]) {
                Passenger p1 = new Passenger("P1");
                Passenger p2 = new Passenger("P2");
                Thread t1 = new Thread(p1);
                Thread t2 = new Thread(p2);
                if (p1.booked == false && p2.booked == false) {
                        t1.start();
                }
                if (p1.booked == false && p2.booked == false) {
                        t2.start();
                }
                System.out.println(p1.getName()+" "+p1.getSeatAndPNR());
                System.out.println(p2.getName()+" "+p2.getSeatAndPNR());
        }
}
```

1. Identify the critical section in this code.

   √ Lines 9 to 11

   ◯ Lines 26 to 31

   ◯ Lines 32 to 33

   ◯ There is no critical section in this code.

Solution: The same resource, that is seat number 52 in coach number 10, may be booked by both P_1 and P_2 due to lines in 10 to 12 being executed by objects of both passengers.

2. Which of the following is/are NOT possible outcome(s) of the execution of the given code?

   ◯ `P1 0 0`
   `P2 10 52`

   √ `P2 10 52`
   `P1 0 0`

   ◯ `P1 10 52`
   `P2 10 52`

   ◯ `P1 10 0`
   `P2 10 52`

Solution: Option 1 is possible because the main program is also running as a thread. Hence, the main program may execute line 32 before t1 executes line 10 and 11 (before setting the coach_no and seat_no of p1).

Order of execution:
32, 10, 11

Option 3 is possible when both t1 and t2 are executed before main executes line 33.

Order of execution:
9 - 11 for both t1 and t2, then 33

Option 4 is possible when both t1 and t2 are executed before main executes line 32.

Order of execution:
Line 9 of t1 (thread t1)
Line 9 to 11 of t2 (thread t2)
Line 32 (main thread)
Line 14 of t1 (thread t2)
Line 10 of t1 (thread t1)

In whatever the order the threads may get executed, the values of p1 will be printed first. Hence, option 2 will never be the correct output.

3. Consider the Java code given below.

```
class Expression implements Runnable{
    static int a = 1, b = 2;
    static boolean ready_1=false, ready_2=true;
    public void run() {
    }
}
class Expression1 extends Expression {
    public void run() {
        while (Expression.ready_2) {}
        Expression.a = 3 * (Expression.b/2);
        Expression.a = Expression.a * Expression.b;
        Expression.ready_1 = false;
    }
}
class Expression2 extends Expression{
    public void run() {
        while (Expression.ready_1) {}
        Expression.a = 4 * (Expression.b/2);
        Expression.a = Expression.a * Expression.b;
        Expression.ready_2 = false;
    }
}
public class ExpressionEval {
    public static void main(String[] args) {
        Expression expr1 = new Expression1();
        Expression expr2 = new Expression2();
        Thread t1 = new Thread(expr1);
        Thread t2 = new Thread(expr2);
        t1.start();
        t2.start();
        Thread.sleep(10); //try catch removed to simplify the code
    }
}
```

Assuming that the program terminates normally after a certain amount of time, which among the following is/are the possible value/s of `Expression.a` after the execution is completed?

- √ 6
- ◯ 8
- ◯ 12
- ◯ 16

Solution: The codes inside both the `run()` methods are mutually excluded from being interleaved by using the two variables `ready_1` and `ready_2`. Hence, when the code finishes execution, the value computed will be the value obtained by executing run() of Expression 2, followed by executing run() of Expression 1.

4. Consider the code given below.

```
class PrlCls extends Thread{
    private int init;
    public PrlCls(int n){
        init = n;
    }
    public void run(){
        for(int i = init; i <= init + 5; i++){
            System.out.print(i + " ");
            try{
                Thread.sleep(1000);
            }
            catch(InterruptedException e){}
        }
    }
}
public class FClass{
    public static void main(String[] args){
        Thread t1 = new PrlCls(10);
        Thread t2 = new PrlCls(20);
        t1.start();
        for(int i = 30; i < 35; i++){
            System.out.print(i + " ");
        }
        t2.start();
    }
}
```

Choose the correct option regarding the code.

- ◯ It may print 10 to 15, 20 to 25 and 30 to 35 in an interleaved manner.
- ◯ It may print 10 to 15 and 30 to 35 in an interleaved manner. However, 20 to 25 will always be printed after 10 to 15 and 30 to 35 are printed.
- ◯ It may print 10 to 15 and 30 to 35 in an interleaved manner. It may print 30 to 35 and 20 to 25 in an interleaved manner. However, 10 to 15 and 20 to 25 cannot be printed in an interleaved manner.
- √ It may print 10 to 15 and 30 to 35 in an interleaved manner. It may print 10 to 15 and 20 to 25 in an interleaved manner. However, 30 to 35 and 20 to 25 cannot be printed in an interleaved manner.

Solution: Since, the thread `t2` starts after the values 30 to 35 are printed, 30 to 35 and 20 to 25 cannot be printed in an interleaved manner.

5. Consider there are three threads T1, T2, T3 and two shared variables named `a, b` with initial values 1 and 9 respectively in a program.
   Thread T1 will execute a pseudo code segment to assign values to variable `a`
   Thread T2 will execute a pseudo code segment to assign values to variable `b`
   Thread T3 will print "`Sum of a and b is 10`" if `a` + `b` = 10.

   T1's code:

```
forLoop(i=2; i<=5 ;i++){
    P(S3)
    P(S2)
    a = i
    V(S1)
    // LINE 1
}
```

   T2's code:

```
forLoop(j=8; j>=5 ; j--){
    P(S3)
    P(S1)
    b = j
    V(S2)
    // LINE 2
}
```

   T3's code:

```
k = 5
whileLoop(k>0){
     P(S1)
     P(S2)
     if (a + b == 10)
     Then print "Sum of a and b is 10"
     k--
     V(S3)
     // LINE 3
}
```

   Assumptions:

   1. A semaphore can have values ranging from 0 to 2
   2. If a semaphore has value 0 then P() operation can't be used on it.
   3. If a semaphore has value 2 then V() operation can't be used on it.

If this program prints the statement `Sum of a and b is 10` total 5 times on every execution. Then choose the correct set of instructions which should be given in place of LINE 1, 2, 3.

- √ `LINE 1`: V(S1), `LINE 2`: V(S2), `LINE3`: V(S3)
- ◯ `LINE 1`: V(S3), `LINE 2`: V(S2), `LINE 3`: V(S1)
- ◯ `LINE 1`: V(S2), `LINE 2`: V(S3), `LINE 3`: V(S1)
- ◯ `LINE 1`: V(S1), `LINE 2`: P(S2), `LINE 3`: V(S3)

Solution:
We would need at least 3 semaphores for this example.
The code blocks would be as follows.
T1's code:

```
forLoop(i=2; i<=5 ;i++){
    P(S3)
    P(S2)
    a = i
    V(S1)
    V(S1)
}
```

T2's code:

```
forLoop(j=8; j>=5 ; j--){
    P(S3)
    P(S1)
    b = j
    V(S2)
    V(S2)
}
```

T3's code:

```
k = 5
whileLoop(k>0){
    P(S1)
    P(S2)
    if (a + b == 10)
    Then print "Sum of a and b is 10"
    k--
```

```
    V(S3)
    V(S3)
}
```

We will initialize the semaphores as follows
S1 = 2, S2 = 2, S3 = 0. Will be discussed in detail in live session.

6. Consider the following code.

```
class Point implements Runnable{
    static int x, y;
    private String name;
    public Point(String n){
      this.name = n;
    }
    public void updatePoint(int a, int b){
      x = a;
      y = b;
    }
    public void display(){
      System.out.println("("+ x +","+ y + ")");
    }
    public void run(){
      if(name.equals("update")){
        updatePoint(this.x + 5, this.y + 6);
      }
      if(name.equals("display")){
        display();
      }
    }
}
public class PointUser {
    public static void main(String[] args){
        Point p1 = new Point("update");
        Point p2 = new Point("display");
        Thread t1 = new Thread(p1);
        Thread t2 = new Thread(p2);
        t2.start();
        t1.start();
    }
}
```

Which among the following is/are possible scenarios of execution (assuming order of execution within a thread is sequential)?

- √ `updatePoint()` of `t1` after `display()` of `t2`
- √ `display()` of `t1` after `updatePoint()` of `t2`
- ◯ `updatePoint()` of `t1` after `display()` of `t1`
- ◯ `updatePoint()` of `t2` after `display()` of `t2`

Consider the code given below, and answer questions 7 and 8.

```
class Bank{
    int balance = 100;
    void extract(String name,int withdrawal){
        if (balance>=withdrawal){
            System.out.println(name + " has withdrawn "+ withdrawal);
            balance = balance - withdrawal;
        }
        else {
            System.out.println(name+ ", cannot withdraw "+ withdrawal);
            System.out.println("Because current balance is " + balance);
        }
    }
}
class Withdrawal implements Runnable{
    Bank bk;
    String name;
    int money;
    Withdrawal(Bank ob, String name, int money){
        this.bk=ob;
        this.name=name;
        this.money=money;
    }
    public void run(){
        bk.extract(name, money);
    }
}
public class ThreadBank{
    public static void main(String[] args){
        Bank obj = new Bank();
        Thread t1=new Thread(new Withdrawal(obj, "Sun", 20));
        Thread t2=new Thread(new Withdrawal(obj, "Moon", 40));
        Thread t3=new Thread(new Withdrawal(obj, "Earth", 80));
        t1.start();
        t2.start();
        t3.start();
    }
}
```

7. Which among the following is/are possible output/s of the given program?

   √ Earth has withdrawn 80
   Sun has withdrawn 20
   Moon has withdrawn 40

   √ Sun has withdrawn 20
   Moon has withdrawn 40
   Earth cannot withdraw 80
   Because current balance is 40

   √ Earth has withdrawn 80
   Moon cannot withdraw 40
   Because current balance is 20
   Sun has withdrawn 20

   √ Moon has withdrawn 40
   Sun has withdrawn 20
   Earth has withdrawn 80

8. In the given code, suppose you change the statements

```
t1.start();
t2.start();
t3.start();
```

   to

```
t1.run();
t2.run();
t3.run();
```

   Then, which among the following is/are the possible output/s?

   ◯ Earth has withdrawn 80
   Sun has withdrawn 20
   Moon has withdrawn 40

   √ Sun has withdrawn 20
   Moon has withdrawn 40
   Earth cannot withdraw 80
   Because current balance is 40

   ◯ Earth has withdrawn 80
   Moon cannot withdraw 40
   Because current balance is 20
   Sun has withdrawn 20

   ◯ Moon has withdrawn 40
   Sun has withdrawn 20
   Earth has withdrawn 80

BSCCS2005: Practice with Solutions
Week 11

1. Consider the code given below.

```
public class PrlSequence1 extends Thread{
    public void run(){
        try {
            for(int i = 11; i <= 20; i++){
                if(isInterrupted())
                    break;
                System.out.print(i + " ");
                Thread.sleep(500);
            }
        }catch(InterruptedException e) {}
    }
}

public class PrlSequence2 extends Thread{
    Thread other;
    PrlSequence2(Thread t){
        other = t;
    }
    public void run(){
        try {
            for(int i = 21; i <= 30; i++){
                System.out.print(i + " ");
                if(i == 25)
                    other.interrupt();
                Thread.sleep(500);
            }
        }
        catch(InterruptedException e) {}
    }
}

public class FClass{
    public static void main(String[] args) throws InterruptedException{
        Thread th1 = new PrlSequence1();
        Thread th2 = new PrlSequence2(th1);
        th1.start();
        th2.start();
    }
}
```

Choose the correct option regarding the code.

- ◯ Thread `th1` prints 11 to 20, and thread `th2` prints 21 to 30 in an interleaved way.
- ◯ The thread `th1` first prints 11 to 20, and followed by the thread `th2` prints 21 to 30.
- ◯ The thread `th1`, and the thread `th2` initially may print the numbers in an interleaved way. However, the thread `th2` gets terminated after printing 25
- √ The thread `th1`, and the thread `th2` initially may print the numbers in an interleaved way. However, the thread `th2` after printing 25, causes thread `th1` to terminate.

Solution:

2. Consider the following code which tries to simulate a producer-consumer relationship on cakes.

```
class CakeOperation {
    int available_cake_count=0;

    synchronized void eatCake() {
        available_cake_count--;
        System.out.println("Ate one cake. Cakes left: "+ available_cake_count);
    }
    synchronized void makeCake() {
        available_cake_count++;
        System.out.println("Made one cake. Cakes left: "+ available_cake_count);
    }
}
class Baker implements Runnable {
    CakeOperation obj;
    Baker(CakeOperation o) {
        obj = o;
        Thread Producer = new Thread(this);
        Producer.start();
    }
    public void run() {
        int cake_number = 1;
        while(cake_number <= 2) {
            obj.makeCake();
            cake_number++;
        }
    }
}
class Consumer implements Runnable {
    CakeOperation obj;
    Consumer(CakeOperation o) {
        obj = o;
        Thread consumer = new Thread(this);
        consumer.start();
    }
    public void run() {
        int iteration = 0;
        while(iteration < 2) {
            obj.eatCake();
            iteration++;
        }
    }
```

```
}
public class Test {
    public static void main(String args[]) {
        CakeOperation obj  = new CakeOperation();
        Baker b1 = new Baker(obj);
        Consumer c1 = new Consumer(obj);
    }
}
```

Which of the following options is/are NOT possible result/s of the above code?

- √ Made one cake. Cakes left: 1
  Made one cake. Cakes left: 1
  Ate one cake. Cakes left: 0
  Ate one cake. Cakes left: 0
- ◯ Made one cake. Cakes left: 1
  Ate one cake. Cakes left: 0
  Made one cake. Cakes left: 1
  Ate one cake. Cakes left: 0
- ◯ Made one cake. Cakes left: 1
  Ate one cake. Cakes left: 0
  Ate one cake. Cakes left: -1
  Made one cake. Cakes left: 0
- ◯ Made one cake. Cakes left: 1
  Made one cake. Cakes left: 2
  Ate one cake. Cakes left: 1
  Ate one cake. Cakes left: 0

Solution: Detailed sol goes here

3. Consider the code given below.

```
class Bank{
    int total = 100;
    synchronized void extract(String name,int withdrawal){
        if (total >= withdrawal){
            System.out.println(name + " has withdrawn " + withdrawal);
            total = total - withdrawal;
        }
        else {
            System.out.println(name + " can not withdraw "+ withdrawal);
            System.out.println("Current balance is: " + total);
        }
    }
}
class Withdrawal implements Runnable{
    Bank bk;
    String name;
    int money;
    Withdrawal(Bank b, String n, int m){
        this.bk = b;
        this.name = n;
        this.money = m;
    }
    public void run(){
        bk.extract(name, money);
    }
}
public class Test {
    public static void main(String[] args){
        Bank obj = new Bank();
        Thread t1 = new Thread(new Withdrawal(obj, "Sun", 20));
        Thread t2 = new Thread(new Withdrawal(obj, "Moon", 40));
        Thread t3 = new Thread(new Withdrawal(obj, "Earth", 80));
        t1.start();
        t2.start();
        t3.start();
    }
}
```

What is/are the possible output/s?

√ `Sun has withdrawn 20`
`Earth has withdrawn 80`
`Moon can not withdraw 40`
`Current balance is: 0`

√ `Sun has withdrawn 20`
`Moon has withdrawn 40`
`Earth can not withdraw 80`
`Current balance is: 40`

◯ `Moon has withdrawn 40`
`Sun has withdrawn 20`
`Earth has withdrawn 80`

◯ This program generates `IllegalMonitorStateException` at runtime.

Solution: The non `static synchronization` provides an object level lock and allows only 1 thread at a time to enter the critical region. An object can have multiple threads but the sensitive area can only be accessed by 1 thread at a time.

4. Consider the code given below.

```
class ThExample extends Thread{
    int a = 1, b = 1, c = 0;
    public void run(){
       synchronized(this){
           for(int n = 0; n < 5; n++){
               c = a + b;
               a = b;
               b = c;
            }
            this.notify();
        }
    }
}

public class FClass{
    public static void main(String[] args) throws InterruptedException{
        ThExample t = new ThExample();
        t.start();
        synchronized(t){
            t.wait();
            System.out.println(t.c);
        }

     }
}
```

Choose the correct option regarding the given code.

- √ This code may generate the output
  13
- ◯ This code may generate the output
  8
- ◯ This code may generate the output
  0
- ◯ This code may generate any Fibonacci number within the closed interval [2, 13].

5. Consider the code given below.

```
import java.util.concurrent.locks.*;
class Bank{
    ReentrantLock lck = new ReentrantLock();
    int balance = 100;
    void extract(String name,int withdrawal){
        lck.lock();
        try{
            if (balance>=withdrawal){
                System.out.println(name + " has withdrawn " + withdrawal);
                balance = balance - withdrawal;
            }
            else {
                System.out.println(name + " cannot withdraw " + withdrawal);
                System.out.println("Because current balance is " + balance);
            }
        }
        finally{
            lck.unlock();
        }
    }
}
class Withdrawal implements Runnable{
    Bank l_obj;
    String name;
    int money;
    Withdrawal(Bank o, String n, int m){
        this.l_obj = o;
        this.name = n;
        this.money = m;
    }
    public void run(){
        l_obj.extract(name, money);
    }
}
public class FClass{
    public static void main(String[] args){
        Bank obj = new Bank();
        Thread t1 = new Thread(new Withdrawal(obj, "Sun", 20));
        Thread t2 = new Thread(new Withdrawal(obj, "Moon", 40));
        Thread t3 = new Thread(new Withdrawal(obj, "Earth", 80));
        t1.start();
        t2.start();
```

```
        t3.start();
    }
}
```

What is/are the possible output/s?

- √ Sun has withdrawn 20
  Earth has withdrawn 80
  Moon cannot withdraw 40
  Because current balance is 0
- ◯ Earth has withdrawn 80
  Moon has withdrawn 40
  Sun has withdrawn 20
- √ Sun has withdrawn 20
  Moon has withdrawn 40
  Earth cannot withdraw 80
  Because current balance is 40
- ◯ Moon has withdrawn 40
  Sun has withdrawn 20
  Earth has withdrawn 80

6. Consider the code given below.

```
public class Test{
    public static void main(String[] args){
        Thread t=Thread.currentThread();
        t.interrupt();
        System.out.println(Thread.interrupted());
        while(t.isInterrupted()){
            System.out.println("Infinite");
        }
    }
}
```

What will the output be?

- √ `true`
- ◯ `false`
- ◯ `true`
  `Infinite`
- ◯ This program prints `true` followed by infinite loops printing `Infinite`.

Solution: The `interrupted` method prints the interruption status and change it to `false` if the status is `true`.

7. Consider the code given below.

```
import java.util.*;
class PrlTest extends Thread{
    Map<String, Integer> icMap;
    public PrlTest(Map<String, Integer> ic){
        this.icMap = ic;
    }
    public void run(){
        icMap.put("D", 4);
    }
}
public class FClass{
    public static void main (String[] args) throws InterruptedException{
        Thread t1 = Thread.currentThread();
        Map<String, Integer> icMap = new LinkedHashMap<String, Integer>();
        String[] str = {"A", "B", "C"};
        Integer[] arr = {1, 2, 3};
        for(int i = 0; i < str.length; i++){
            icMap.put(str[i], arr[i]);
        }
        PrlTest t2 = new PrlTest(icMap);
        t2.start();
        t2.join();
        for(Map.Entry m : icMap.entrySet()){
            System.out.println(m.getKey() + " => "+ m.getValue());
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ This program prints
  `A => 1`
  `B => 2`
  `C => 3`
- ◯ This program results in a deadlock.
- √ This program prints
  `A => 1`
  `B => 2`
  `C => 3`
  `D => 4`
- ◯ This program prints all 4 elements: `A => 1`, `B => 2`, `C => 3` and `D => 4`. However, the order of the elements can't be undetermined.

8. Consider the code given below.

```
import java.util.*;
class Example extends Thread{
    Map mp;
    Example(Map m){
        this.mp = m;
    }
    public void run(){
        mp.put("D",4);
    }
}
public class Test{
    public static void main (String[] args){
        Map<String, Integer> map = new LinkedHashMap();
        String[] str = {"A", "B", "C"};
        Integer[] arr = {1, 2, 3};
        for(int i = 0;i < str.length; i++){
           map.put(str[i],arr[i]);
        }
        Example t = new Example(map);
        t.start();
        Set s = map.entrySet();
        Iterator itr = s.iterator();
        while(itr.hasNext()){
           Map.Entry m = (Map.Entry)itr.next();
           System.out.println(m.getKey() + " => " + m.getValue());
        }
    }
}
```

Which of the following is NOT true about the given code.

- ◯ This program may generate `ConcurrentModificationException`.
- ◯ This program may generate the output

  ```
  A => 1
  B => 2
  C => 3
  ```

- ◯ This program may generate the output

  ```
  A => 1
  B => 2
  C => 3
  D => 4
  ```

√ This program may generate the output

```
D => 4
A => 1
B => 2
C => 3
```

9. In the following program, there are two user-defined threads named `Dijkstra` and `Turing`. The program prints different possible thread states that each of these threads can be in, during the execution of program.

```
import java.util.*;
public class ThreadState implements Runnable   {

    public synchronized void run() {
        try {
            Thread.sleep(800);
        }catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName()+
                        " is inside run method but not sleeping");
    }
    public static void main(String args[]) throws InterruptedException  {
        ThreadState obj = new ThreadState();
        Thread t1= new Thread(obj,"Dijkstra");
        Thread t2= new Thread(obj,"Turing");
        Thread.currentThread().setPriority(10);
        System.out.println("Dijkstra before calling start(): "+t1.getState());
        System.out.println("Turing before calling start(): "+t2.getState());
        t1.start();
        t2.start();
        System.out.println("Dijkstra after calling start(): "+t1.getState());
        System.out.println("Turing after calling start(): "+t2.getState());

        Thread.sleep(200);
        System.out.println("Dijkstra before joining: "+t1.getState());
        System.out.println("Turing before joining: "+t2.getState());

        t1.join();
        t2.join();
        System.out.println("Dijkstra after joining: "+t1.getState());
        System.out.println("Turing after joining: "+t2.getState());
    }
}
```

Note:
1. A thread remains in `TIMED_WAITING` state when it is sleeping.
2. A thread is in `BLOCKED` state when it is waiting for the object lock to be released by some other thread which is using that object.

Choose all the options corresponding to an output, that would NEVER occur as a result of this program.

√ Dijkstra before calling start(): NEW
Turing before calling start(): NEW
Dijkstra after calling start(): RUNNABLE
Turing after calling start(): RUNNABLE
Dijkstra before joining: TIMED_WAITING
Turing before joining: TIMED_WAITING
Turing is inside run method but not sleeping
Dijkstra is inside run method but not sleeping
Dijkstra after joining: TERMINATED
Turing after joining: TERMINATED

◯ Dijkstra before calling start(): NEW
Turing before calling start(): NEW
Dijkstra after calling start(): RUNNABLE
Turing after calling start(): RUNNABLE
Dijkstra before joining: TIMED_WAITING
Turing before joining: BLOCKED
Dijkstra is inside run method but not sleeping
Turing is inside run method but not sleeping
Dijkstra after joining: TERMINATED
Turing after joining: TERMINATED

◯ Dijkstra before calling start(): NEW
Turing before calling start(): NEW
Dijkstra after calling start(): RUNNABLE
Turing after calling start(): RUNNABLE
Dijkstra before joining: BLOCKED
Turing before joining: TIMED_WAITING
Turing is inside run method but not sleeping
Dijkstra is inside run method but not sleeping
Dijkstra after joining: TERMINATED
Turing after joining: TERMINATED

◯ Dijkstra before calling start(): NEW
Turing before calling start(): NEW
Dijkstra after calling start(): TIMED_WAITING
Turing after calling start(): BLOCKED
Dijkstra before joining: TIMED_WAITING
Turing before joining: BLOCKED
Dijkstra is inside run method but not sleeping
Turing is inside run method but not sleeping
Dijkstra after joining: TERMINATED
Turing after joining: TERMINATED

◯ `Dijkstra before calling start(): NEW`
`Turing before calling start(): NEW`
`Dijkstra after calling start(): TIMED_WAITING`
`Turing after calling start(): RUNNABLE`
`Dijkstra before joining: TIMED_WAITING`
`Turing before joining: BLOCKED`
`Dijkstra is inside run method but not sleeping`
`Turing is inside run method but not sleeping`
`Dijkstra after joining: TERMINATED`
`Turing after joining: TERMINATED`

Solution:
A thread is in `TIMED_WAITING` state when it is sleeping, the only part of code where either `Dijkstra` or `Turing` can execute a `Thread.sleep()` instruction, is inside the `run` method. Now, in the first option both the threads are together in `TIMED_WAITING` state, which means both have been inside the `run` method together and thus mutual exclusion couldn't be achieved, but `run` method is synchronized in this example and hence it must guarantee mutual exclusion on an object of `ThreadState` class.

Thus we reach at a contradiction which clearly indicates that the output given in option 1 can never occur.

BSCCS2005: Practice with Solutions
Week 12

1. Consider the code given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class FClass implements ActionListener{
    JFrame frm;
    JTextField inText, outText;
    JButton cmBtn, ftBtn;
    JLabel lblIn, lblOut;
    JPanel inputPanel, outputPanel, btnPanel;
    FClass(){
        frm = new JFrame("Height Converter");
        frm.setSize(340, 140);
        lblIn = new JLabel("Height (inch)");
        inText = new JTextField(10);
        lblOut = new JLabel("Output");
        outText = new JTextField(10);
        cmBtn = new JButton("Convert to CM");
        ftBtn = new JButton("Convert to FT");
        cmBtn.setActionCommand("cm");
        ftBtn.setActionCommand("ft");
        cmBtn.addActionListener(this);
        ftBtn.addActionListener(this);

        //add inputPanel, outputPanel and btnPanel to
        //the "North", "Center" and "Bottom" of the JFrame

        //add lblIn and inText to inputPanel
        //add lblOut and outText to outputPanel
        //add cmBtn and ftBtn to btnPanel
        frm.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        //code-segment: to be added
    }
    public static void main(String[] args){
        new FClass();
    }
}
```

Choose the option that has correct implementation of the method `actionPerformed` such that it converts a height given in inches to cm by clicking the button labelled "Convert to CM" and to feet (ft) by clicking the button labelled "Convert to FT".

○
```
double in = Double.parseDouble(inText.getText());
JButton btn = e.getSource();
double out = 0.0;
if(btn.equals(ftBtn))
    out = in * 0.0833333;
else if(btn.equals(cmBtn))
    out = in * 2.54;
outText.setText(out + "");
```

√
```
double in = Double.parseDouble(inText.getText());
JButton btn = (JButton)e.getSource();
double out = 0.0;
if(btn.equals(ftBtn))
    out = in * 0.0833333;
else if(btn.equals(cmBtn))
    out = in * 2.54;
outText.setText(out + "");
```

○
```
double in = Double.parseDouble(inText.getText());
String s = e.getActionCommand();
double out = 0.0;
if(s.equals(ftBtn))
    out = in * 0.0833333;
else if(s.equals(cmBtn))
    out = in * 2.54;
outText.setText(out + "");
```

√
```
double in = Double.parseDouble(inText.getText());
String s = e.getActionCommand();
double out = 0.0;
if(s.equals("ft"))
    out = in * 0.0833333;
else if(s.equals("cm"))
    out = in * 2.54;
outText.setText(out + "");
```

2. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class GUITest extends JFrame implements WindowListener{
    JLabel label;
    Container contentPane;
    String str="";
    public GUITest() {
        label=new JLabel();
        addWindowListener(this);
        contentPane = this.getContentPane();
        contentPane.add(label);
        setVisible(true);
        setSize(700,200);
    }
    public void windowOpened(WindowEvent e) {
        str+="Window Opened ";
        label.setText(str);
    }
    public void windowClosing(WindowEvent e) {
        str+="Window Closing ";
        label.setText(str);
    }
    public void windowClosed(WindowEvent e) {
    }
    public void windowIconified(WindowEvent e) {
        str+="Window Iconified ";
        label.setText(str);
    }
    public void windowDeiconified(WindowEvent e) {
        str+="Window Deiconified ";
        label.setText(str);
    }
    public void windowActivated(WindowEvent e) {
        str+="Window Activated";
        label.setText(str);
    }
    public void windowDeactivated(WindowEvent e) {
        str+="Window Deactivated ";
        label.setText(str);
    }
    public static void main(String[] args) {
```

```
        new GUITest();
    }
}
```

Method description:

`void windowOpened(WindowEvent e):`
Invokes the first time a window is made visible.

`void windowClosing(WindowEvent e):`
Invoked when the user attempts to close the window.

`void windowClosed(WindowEvent e):`
Invoked when a window has been closed.

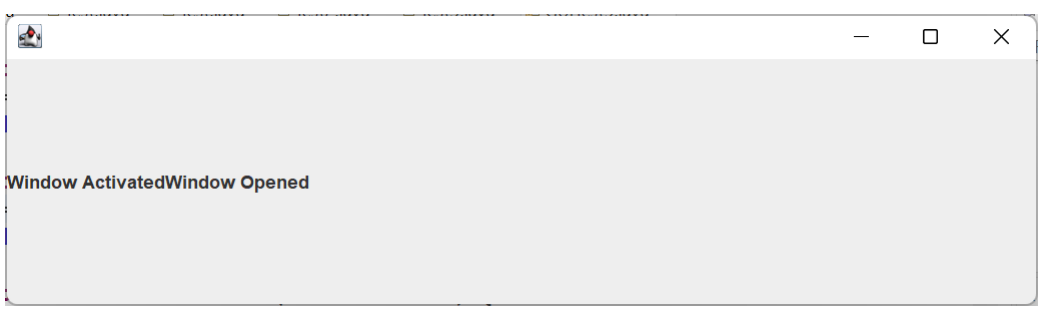
`void windowIconified(WindowEvent e):`
Invoked when a window is changed from a normal to a minimized state.

`void windowDeiconified(WindowEvent e):`
Invoked when a window is changed from a minimized to a normal state.

`void windowActivated(WindowEvent e):`
Invoked when the window is set to be the active window.

`void windowDeactivated(WindowEvent e):`
Invoked when a window is no longer the active window.

Which of the following possible GUIs generated by the given code:



√

√ 

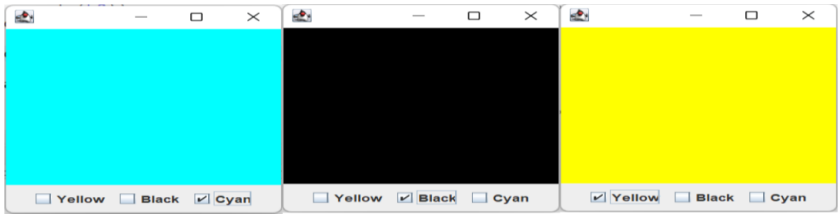
○ 

○ All of the above.

Solution:

3. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class GUITest extends JFrame implements ActionListener{
    JCheckBox b1,b2,b3;
    JPanel panel1,panel2;
    public GUITest(){
        panel1=new JPanel();
        panel2=new JPanel();
        b1=new JCheckBox("Yellow");
        b2=new JCheckBox("Black");
        b3=new JCheckBox("Cyan");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        panel1.add(b1);
        panel1.add(b2);
        panel1.add(b3);
        add(panel1,"South");
        add(panel2,"Center");
        setVisible(true);
        setSize(400,400);
    }
    public void actionPerformed(ActionEvent e) {
        --------------------------------------
        *************CODE SEGMENT*************
        --------------------------------------
    }
    public static void main(String[] args){
        new GUITest();
    }
}
```



Choose the correct code segment inside method `actionPerformed()` such that whenever either of the three checkboxes (Yellow/Black/Cyan) is clicked on panel1, background color of panel2 changes accordingly.

√
```
if(e.getSource().equals(b1))
    panel2.setBackground(Color.yellow);
if(e.getSource().equals(b2))
    panel2.setBackground(Color.black);
if(e.getSource().equals(b3))
    panel2.setBackground(Color.cyan);
```

◯
```
if(e.getSource().equals(b1))
    panel1.setBackground(Color.yellow);
if(e.getSource().equals(b2))
    panel1.setBackground(Color.black);
if(e.getSource().equals(b3))
    panel1.setBackground(Color.cyan);
```

◯
```
if(e.equals(b1))
    panel2.setColor(Color.yellow);
if(e.equals(b2))
     panel2.setColor(Color.black);
if(e.equals(b3))
     panel2.setColor(Color.cyan);
```

◯
```
if(e.getSource().equals(b1))
    panel2.setColor(Color.yellow);
if(e.getSource().equals(b2))
    panel2.setColor(Color.black);
if(e.getSource().equals(b3))
    panel2.setColor(Color.cyan);
```

Solution:

4. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class GUITest extends JFrame implements KeyListener{
    JLabel label;
    Container contentPane;
    public GUITest() {
        label=new JLabel();
        addKeyListener(this);
        contentPane = this.getContentPane();
        contentPane.add(label);
        setVisible(true);
        setSize(200,200);
    }
    public void keyTyped(KeyEvent e) {
        label.setText("KeyTyped");
    }
    public void keyPressed(KeyEvent e) {
        label.setText("keyPressed");
    }
    public void keyReleased(KeyEvent e) {
        label.setText("keyReleased");
    }
    public static void main(String[] args) {
        new GUITest();
    }
}
```

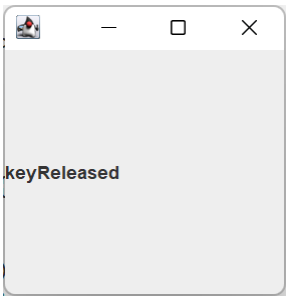
Method description:

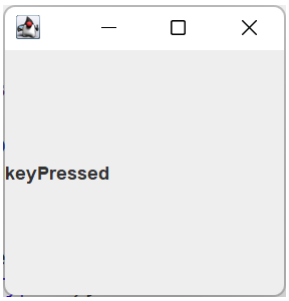
`void keyTyped(KeyEvent e):`
Invoked when a key has been typed.

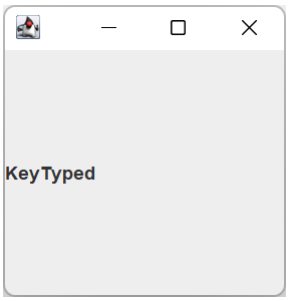
`void keyPressed(KeyEvent e):`
Invoked when a key has been pressed.

`void keyReleased(KeyEvent e):`
Invoked when a key has been released

Which of the following possible GUIs generated by the given code:

○ 

○ 

○ 

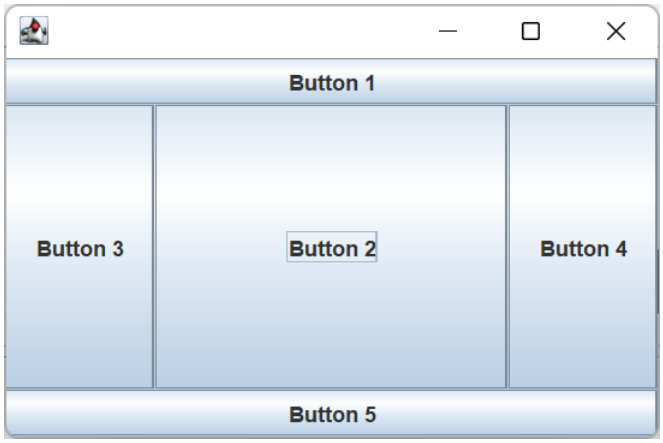
√ All of the above.

Solution:

5. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
public class GUITest extends JFrame{
    JButton b1,b2,b3,b4,b5;
    LayoutManager manager;
    public GUITest(){
        //LINE 1
        setLayout(manager);
        b1=new JButton("Button 1");
        b2=new JButton("Button 2");
        b3=new JButton("Button 3");
        b4=new JButton("Button 4");
        b5=new JButton("Button 5");
        add(b1,"North");
        add(b2,"Center");
        add(b3,"West");
        add(b4,"East");
        add(b5,"South");
        setVisible(true);
        setSize(400,400);
    }
    public static void main(String[] args){
        new GUITest();
    }
}
```

Choose the correct option such that the code produces the output given below:



√ `manager=new BorderLayout();`

◯ `manager=new BorderLayout(6);`

- ◯ `manager=new FlowLayout();`
- ◯ `manager=new GridLayout();`

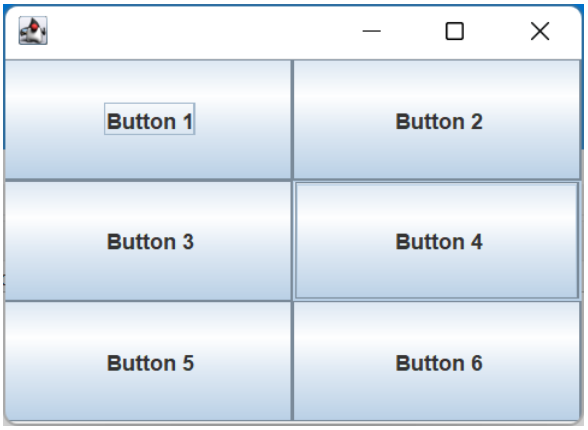
Solution: In above program all buttons are arranged to the frame/window using BorderLayout.

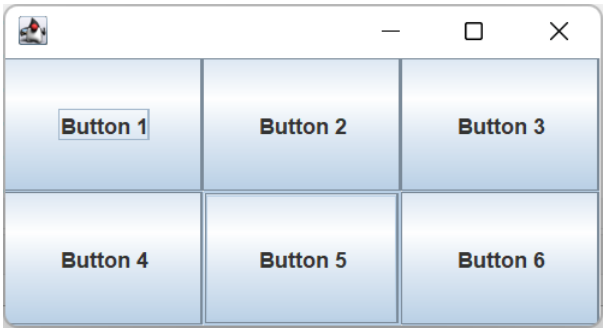
6. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
public class GUITest extends JFrame{
    JButton b1,b2,b3,b4,b5,b6;
    LayoutManager manager;
    public GUITest(){
        manager=new GridLayout(3,2);
        setLayout(manager);
        b1=new JButton("Button 1");
        b2=new JButton("Button 2");
        b3=new JButton("Button 3");
        b4=new JButton("Button 4");
        b5=new JButton("Button 5");
        b6=new JButton("Button 6");
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);
        setVisible(true);
        setSize(400,400);
    }
    public static void main(String[] args){
        new GUITest();
    }
}
```

What will the output be?

○ 





Solution: The program uses GridLayout with 2 rows and 2 columns to arrange the buttons.