Question: 1:
Write a programe do to demonstrate the use of volatile keyword.

```java
package Assignment;

import java.util.Scanner;
class Volatile extends Thread{
    private volatile boolean running = true;
    @Override
    public void run() {
        while (running){
            System.out.println("Hello");
            try{
                Thread.sleep((100));
            } catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }
    public void shutDown(){
        running = false;
    }
}
public class Ques1 {
    public static void main(String[] args) {
        Volatile obj = new Volatile();
        obj.start();
        System.out.println("Press Enter To Stop");
        Scanner in = new Scanner(System.in);
        in.nextLine();

        obj.shutDown();
    }
}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Press Enter To Stop
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello

Process finished with exit code 0
```

Question: 2:
Write a program to create a thread using Thread class and Runnable interface each.

```java
package Assignment;

class Runner1 implements Runnable{
    @Override
    public void run() {
        for (int i = 0; i <10 ; i++) {
            System.out.println("Hello Runnable " + i);
            try {
                Thread.sleep( millis: 100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Runner extends Thread{
    @Override
    public void run() {
        for (int i = 0; i <10 ; i++) {
            System.out.println("Hello Thread " + i);
            try {
                Thread.sleep( millis: 100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```java
public class Ques2 {
    public static void main(String[] args) {
        Runner obj = new Runner();
        Thread obj1 =  new Thread(new Runner1());
        obj.start();
        obj1.start();
    }

}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Hello Runnable 0
Hello Thread 0
Hello Runnable 1
Hello Thread 1
Hello Runnable 2
Hello Thread 2
Hello Runnable 3
Hello Thread 3
Hello Thread 4
Hello Runnable 4
Hello Runnable 5
Hello Thread 5
Hello Runnable 6
Hello Thread 6
Hello Runnable 7
Hello Thread 7
Hello Thread 8
Hello Runnable 8
Hello Thread 9
Hello Runnable 9

Process finished with exit code 0
```

Question: 3:
Write a program using synchronization block and synchronization method

```java
package Assignment;

class SyncMethod {
    private int count = 0;
    public synchronized void increment(){ count++; }
    public void doWork(){
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i <10000; i++) {
                    increment();
                }
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 10000; i++) {
                    increment();
                }
            }
        });
        t1.start();
        t2.start();
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

```
        System.out.println("Count = " + count);
    }
}
class SyncBlock {
    private int count = 0;

    private  Object lock = new Object();
    public void increment(){
        synchronized (lock) {
            count++;
        }
    }

    public void doWork(){
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i <10000; i++) {
                    increment();
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 10000; i++) {
                    increment();
                }
            }
```

```
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 10000; i++) {
                    increment();
                }
            }
        });
        t1.start();
        t2.start();
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Count = " + count);
    }
}

public class Ques3 {
    public static void main(String[] args) {
        SyncMethod obj = new SyncMethod();
        obj.doWork();
        SyncBlock obj2 = new SyncBlock();
        obj2.doWork();
    }

}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Count = 20000
Count = 20000

Process finished with exit code 0
```

Question: 4:
Write a program to create a Thread pool of 2 threads where one Thread will print even numbers and other will print odd numbers.

```java
package Assignment;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

class Even implements Runnable{
    int number;

    Even(int number){
        this.number = number;
    }
    @Override
    public void run() {
        for (int i = 2; i < number; i+=2) {
            System.out.println("Even Thread" + i);
        }
    }
}

class Odd implements Runnable{
    int number;

    Odd(int number){
        this.number = number;
    }
    @Override
    public void run() {
        for (int i = 1; i < number; i+=2) {
            System.out.println("Odd Thread" + i);
        }
    }
}
```

```java
public class Ques4 {

    public static void main(String[] args) {
        ExecutorService executor  = Executors.newFixedThreadPool( nThreads: 2);
        executor.submit(new Even( number: 20));
        executor.submit(new Odd( number: 20));

        executor.shutdown();
        try{
            executor.awaitTermination( timeout: 1, TimeUnit.DAYS);
        } catch (InterruptedException e){
            e.printStackTrace();
        }

    }

}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Even Thread2
Even Thread4
Even Thread6
Even Thread8
Even Thread10
Even Thread12
Even Thread14
Even Thread16
Even Thread18
Odd Thread1
Odd Thread3
Odd Thread5
Odd Thread7
Odd Thread9
Odd Thread11
Odd Thread13
Odd Thread15
Odd Thread17
Odd Thread19

Process finished with exit code 0
```

Question: 5:
Write a program to demonstrate wait and notify methods.

```java
package Assignment;

import java.util.Scanner;

class Processor {
    public void produce() throws InterruptedException {
        synchronized (this) {
            System.out.println("Producer Thread Running");
            wait();
            System.out.println("Resumed Producer");
        }
    }

    public void consume() throws InterruptedException {
        Scanner in = new Scanner(System.in);
        Thread.sleep( millis: 2000);

        synchronized (this){
            System.out.println("Waiting for Return Key");
            in.nextLine();
            System.out.println("Return Key Pressed");
            notify();
            Thread.sleep( millis: 5000);
        }
    }
}
```

```java
public class Ques5 {
    public static void main(String[] args){
        Processor proc = new Processor();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    proc.produce();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    proc.consume();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        t1.start();
        t2.start();
```

```
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Producer Thread Running
Waiting for Return Key

Return Key Pressed
Resumed Producer

Process finished with exit code 0
```

Question: 6:
Write a program to demonstrate sleep and join methods.

```java
public class Ques6 {
    static int count = 0;
    public static void main(String[] args) {
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep( millis: 5000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                count++;
            }
        });

        t1.start();
        try {
            t1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Count is:"+count);

    }
}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Count is:1

Process finished with exit code 0
```

Question: 7:
Run a task with the help of callable and store it's result in the Future.

```java
package Assignment;

import java.util.Random;
import java.util.concurrent.*;

public class Ques7 {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newCachedThreadPool();
        Future<Integer> future =  executor.submit(new Callable<Integer>(){
            @Override
            public Integer call() throws Exception {
                Random rand = new Random();
                System.out.println("Process Started");
                Thread.sleep( millis: 1000);
                System.out.println("Process Ends");
                return rand.nextInt( bound: 100);
            }
        });

        executor.shutdown();

        try {
            executor.awaitTermination( timeout: 1,TimeUnit.DAYS);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }


         try {
             System.out.println("Count :" + future.get());
         } catch (InterruptedException e) {
             e.printStackTrace();
         } catch (ExecutionException e) {
             e.printStackTrace();
         }
    }
}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Process Started
Process Ends
Count :38

Process finished with exit code 0
```

Question: 8:
Write a program to demonstrate the use of semaphore

Question: 9:
Write a program to demonstrate the use of CountDownLatch

Question: 10:
Write a program which creates deadlock between 2 threads

```java
package Assignment;
public class Ques10 {
    public static void main(String[] args) {
        Object lock = new Object();
        Object lock2 = new Object();

        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                synchronized (lock){
                    System.out.println("Lock Accuired By Thread_1");
                    try {
                        Thread.sleep( millis: 1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    System.out.println("Thread_1 Needs Lock2");
                    synchronized (lock2){
                        System.out.println("Lock2 Accuired by Thread_1");
                    }
                }
            }
        });
```

```java
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep( millis: 1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                synchronized (lock2){
                    System.out.println("Lock2 Accuired By Thread_2");
                    try {
                        Thread.sleep( millis: 1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    System.out.println("Thread_2 Needs Lock1");
                    synchronized (lock){
                        System.out.println("Lock Accuired by Thread_2");
                    }
                }
            }
        });
```

```
        t1.start();
        t2.start();
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
/home/ankit/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Lock Accuired By Thread_1
Lock2 Accuired By Thread_2
Thread_1 Needs Lock2
Thread_2 Needs Lock1

Process finished with exit code 130 (interrupted by signal 2: SIGINT)
```