Assignment 8

Q1. c. Using ForkJoinPool :-

```java
import java.util.ArrayList;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveAction;

class Employee{
   /*Selected Employee have current salary, is their current salary in their previous company
    expected_salary, is their salary after the switch,
    fresher_salary_dev , is salary if Employee is fresher than what will be the salary of developer p.a.
    fresher_salary_test, is salary if Employee is fresher and selected for tester role p.a.
    type is the developer or tester role applied for.
    "D" : Developer role
    "T" : Tester role
    we have to compute expected_salary
   */
    double current_salary = 0.0,expected_salary = 0.0;
    int work_exp = 0;

    final double fresher_salary_dev = 10_00000, fresher_salary_test = 8_00000;
    String name,type;


   Employee(double current_salary, String type,String name,int work_exp){
      this.current_salary = current_salary;
      this.type = type;
      this.name = name;
      this.work_exp = work_exp;
   }

   @Override
   public String toString() {
      return "name :"+" "+this.name + ",Role :"+ this.type +",Current_salary :"+ this.current_salary
+",Expected_Salary: "+ this.expected_salary;
   }
}
 class ProcessEmployeeDetails extends RecursiveAction {
   /**
    *
    */
   ArrayList<Employee> employee_details = new ArrayList<>();
   int start,end,index;
   ProcessEmployeeDetails(ArrayList<Employee> employee_details,int start,int end){
      this.employee_details = employee_details;
      this.start = start;
      this.end = end;
      this.index = start;
   }
   private static final long serialVersionUID = 1L;


   public void update() {
      // TODO Auto-generated method stub
         Employee fetched = this.employee_details.get(index);
         if(fetched.type.equals("D")) {
            if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
               fetched.expected_salary = (75 * fetched.current_salary)/100 + fetched.current_salary;
            } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary == 0.0)
{
               fetched.expected_salary = (55 * fetched.current_salary)/100 + fetched.current_salary;
            }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0){
               fetched.expected_salary = (35 * fetched.current_salary)/100 + fetched.current_salary;
            } else {
               fetched.expected_salary = fetched.fresher_salary_dev;
            }
         } else {
            if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
               fetched.expected_salary = (70 * fetched.current_salary)/100 + fetched.current_salary;
            } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary == 0.0)
{
               fetched.expected_salary = (50 * fetched.current_salary)/100 + fetched.current_salary;
            }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0){
               fetched.expected_salary = (30 * fetched.current_salary)/100 + fetched.current_salary;
            } else {
               fetched.expected_salary = fetched.fresher_salary_dev;
            }
         }
      }

   @Override
   protected void compute() {
      // TODO Auto-generated method stub
      if((end - start) == 1) {
         update();
      }
      else {
         int mid = (start + end)/2;
         ProcessEmployeeDetails ped1 = new ProcessEmployeeDetails(employee_details,start,mid);
         ProcessEmployeeDetails ped2 = new ProcessEmployeeDetails(employee_details,mid,end);
         invokeAll(ped1,ped2);
      }
   }
}
public class Threading {
      public static void main(String... gaurav) throws Exception{
         //Creating employee list that got selected.
         ArrayList<Employee> employee_details = new ArrayList<>();



         Employee emp1 = new Employee(0.0,"D","Gaurav Chaudhary",0);
```

```
        Employee emp2 = new Employee(15_00000.0,"D","Gajodhar",12);
        Employee emp3 = new Employee(4_00000.0,"T","Puneet Saini",24);
        Employee emp4 = new Employee(8_00000.0,"D","Gajraj Patra",36);
        Employee emp5 = new Employee(12_00000,"D","Sumit Patidar",6);
        Employee emp6 = new Employee(11_00000,"D","Preeti Chauchan",48);
        Employee emp7 = new Employee(0.0,"D","Garima Garg",0);
        Employee emp8 = new Employee(7_00000,"T","Jatin R B",120);
        Employee emp9 = new Employee(8_00000,"D","Ravi L sahu",24);
        Employee emp10 = new Employee(0.0,"D","Brajesh B",0);

        employee_details.add(emp1);
        employee_details.add(emp2);
        employee_details.add(emp3);
        employee_details.add(emp4);
        employee_details.add(emp5);
        employee_details.add(emp6);
        employee_details.add(emp7);
        employee_details.add(emp8);
        employee_details.add(emp9);
        employee_details.add(emp10);

        ProcessEmployeeDetails ped = new
        ProcessEmployeeDetails(employee_details,0,employee_details.size());

        System.out.println("Before Processing ..."+Thread.currentThread());

        for(int i = 0 ; i < employee_details.size();i++) {
           System.out.println(employee_details.get(i));
        }


        ForkJoinPool pool= ForkJoinPool.commonPool();
        pool.invoke(ped);


        System.out.println("After Processing ..."+pool.getPoolSize());

        for(int i = 0 ; i < employee_details.size();i++) {
           System.out.println(employee_details.get(i));
        }
     }
}
```

Output :-

```
Before Processing ...Thread[main,5,main]
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 0.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 0.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 0.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 0.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 0.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 0.0
After Processing ...3
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 2025000.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 600000.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 1000000.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 1705000.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 1190000.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
```

Q1 c Using Runnable

```java
import java.util.ArrayList;

class Employee{
   /*Selected Employee have current salary, is their current salary in their previous company
    expected_salary, is their salary after the switch,
    fresher_salary_dev , is salary if Employee is fresher than what will be the salary of developer p.a.
    fresher_salary_test, is salary if Employee is fresher and selected for tester role p.a.
    type is the developer or tester role applied for.
    "D" : Developer role
    "T" : Tester role
    we have to compute expected_salary
   */
    double current_salary = 0.0,expected_salary = 0.0;
    int work_exp = 0;

    final double fresher_salary_dev = 10_00000, fresher_salary_test = 8_00000;
    String name,type;


   Employee(double current_salary, String type,String name,int work_exp){
      this.current_salary = current_salary;
      this.type = type;
      this.name = name;
      this.work_exp = work_exp;
   }

   @Override
   public String toString() {
      return "name :"+" "+this.name + ",Role :"+ this.type +",Current_salary :"+ this.current_salary
+",Expected_Salary: "+ this.expected_salary;
   }
}
 class ProcessEmployeeDetails implements Runnable {
   /**
```

```java
     *
     */
    ArrayList<Employee> employee_details = new ArrayList<>();
    int start,end,index;
    ProcessEmployeeDetails(ArrayList<Employee> employee_details){
        this.employee_details = employee_details;
    }

    @Override
    public void run() {
        // TODO Auto-generated method stub
            for(int i = 0 ;i < employee_details.size();i++){
                Employee fetched = employee_details.get(i);
                if(fetched.type.equals("D")) {
                    if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
                        fetched.expected_salary = (75 * fetched.current_salary)/100 + fetched.current_salary;
                    } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary ==
0.0) {
                        fetched.expected_salary = (55 * fetched.current_salary)/100 + fetched.current_salary;
                    }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0)
{
                        fetched.expected_salary = (35 * fetched.current_salary)/100 + fetched.current_salary;
                    } else {
                        fetched.expected_salary = fetched.fresher_salary_dev;
                    }
                } else {
                    if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
                        fetched.expected_salary = (70 * fetched.current_salary)/100 + fetched.current_salary;
                    } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary ==
0.0) {
                        fetched.expected_salary = (50 * fetched.current_salary)/100 + fetched.current_salary;
                    }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0)
{
                        fetched.expected_salary = (30 * fetched.current_salary)/100 + fetched.current_salary;
                    } else {
                        fetched.expected_salary = fetched.fresher_salary_test;
                    }
                }
            }
        }
}
class Threading {
        public static void main(String... gaurav) throws Exception{
            //Creating employee list that got selected.
            ArrayList<Employee> employee_details1 = new ArrayList<>();
            ArrayList<Employee> employee_details2 = new ArrayList<>();
            Employee emp1 = new Employee(0.0,"D","Gaurav Chaudhary",0);
            Employee emp2 = new Employee(15_00000.0,"D","Gajodhar",12);
            Employee emp3 = new Employee(4_00000.0,"T","Puneet Saini",24);
            Employee emp4 = new Employee(8_00000.0,"D","Gajraj Patra",36);
            Employee emp5 = new Employee(12_00000,"D","Sumit Patidar",6);
            Employee emp6 = new Employee(11_00000,"D","Preeti Chauchan",48);
            Employee emp7 = new Employee(0.0,"D","Garima Garg",0);
            Employee emp8 = new Employee(7_00000,"T","Jatin R B",120);
            Employee emp9 = new Employee(8_00000,"D","Ravi L sahu",24);
            Employee emp10 = new Employee(0.0,"D","Brajesh B",0);

            employee_details1.add(emp1);
            employee_details1.add(emp2);
            employee_details1.add(emp3);
            employee_details1.add(emp4);
            employee_details1.add(emp5);
            employee_details2.add(emp6);
            employee_details2.add(emp7);
            employee_details2.add(emp8);
            employee_details2.add(emp9);
            employee_details2.add(emp10);

            ProcessEmployeeDetails ped1 = new ProcessEmployeeDetails(employee_details1);
            ProcessEmployeeDetails ped2 = new ProcessEmployeeDetails(employee_details2);

            Thread t1 = new Thread(ped1);
            Thread t2 = new Thread(ped2);
            System.out.println("Before Processing ..."+Thread.currentThread());

            for(int i = 0 ; i < employee_details1.size();i++) {
                System.out.println(employee_details1.get(i));
            }
            for(int i = 0 ; i < employee_details2.size();i++) {
                System.out.println(employee_details2.get(i));
            }

            t1.start();
            t2.start();


            System.out.println("After Processing ...");

            for(int i = 0 ; i < employee_details1.size();i++) {
                System.out.println(employee_details1.get(i));
            }
            for(int i = 0 ; i < employee_details2.size();i++) {
                System.out.println(employee_details2.get(i));
            }
        }
}
```

Output - >

```
Before Processing ...Thread[main,5,main]
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 0.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 0.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 0.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 0.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 0.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 0.0
After Processing ...
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 0.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 600000.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 1000000.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 1705000.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 1190000.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
```

Q1 a. Extending Thread class

```java
import java.util.ArrayList;

class Employee{
   /*Selected Employee have current salary, is their current salary in their previous company
    expected_salary, is their salary after the switch,
    fresher_salary_dev , is salary if Employee is fresher than what will be the salary of developer p.a.
    fresher_salary_test, is salary if Employee is fresher and selected for tester role p.a.
    type is the developer or tester role applied for.
    "D" : Developer role
    "T" : Tester role
    we have to compute expected_salary
   */
    double current_salary = 0.0,expected_salary = 0.0;
    int work_exp = 0;

    final double fresher_salary_dev = 10_00000, fresher_salary_test = 8_00000;
    String name,type;


   Employee(double current_salary, String type,String name,int work_exp){
      this.current_salary = current_salary;
      this.type = type;
      this.name = name;
      this.work_exp = work_exp;
   }

   @Override
   public String toString() {
      return "name :"+" "+this.name + ",Role :"+ this.type +",Current_salary :"+ this.current_salary
+",Expected_Salary: "+ this.expected_salary;
   }
}
 class ProcessEmployeeDetails extends Thread {
   /**
    *
    */
   ArrayList<Employee> employee_details = new ArrayList<>();
   int start,end,index;
   ProcessEmployeeDetails(ArrayList<Employee> employee_details){
      this.employee_details = employee_details;
   }

   @Override
   public void run() {
      // TODO Auto-generated method stub
          for(int i = 0 ;i < employee_details.size();i++){
              Employee fetched = employee_details.get(i);
              if(fetched.type.equals("D")) {
                  if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
                      fetched.expected_salary = (75 * fetched.current_salary)/100 + fetched.current_salary;
                  } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary == 
0.0) {
                      fetched.expected_salary = (55 * fetched.current_salary)/100 + fetched.current_salary;
                  }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0) 
{
                      fetched.expected_salary = (35 * fetched.current_salary)/100 + fetched.current_salary;
                  } else {
                      fetched.expected_salary = fetched.fresher_salary_dev;
                  }
              } else {
                  if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
                      fetched.expected_salary = (70 * fetched.current_salary)/100 + fetched.current_salary;
                  } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary == 
0.0) {
                      fetched.expected_salary = (50 * fetched.current_salary)/100 + fetched.current_salary;
                  }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0) 
{
                      fetched.expected_salary = (30 * fetched.current_salary)/100 + fetched.current_salary;
                  } else {
                      fetched.expected_salary = fetched.fresher_salary_test;
                  }
              }
          }
      }
}
class Threading {
      public static void main(String... gaurav) throws Exception{
          //Creating employee list that got selected.
          ArrayList<Employee> employee_details1 = new ArrayList<>();
          ArrayList<Employee> employee_details2 = new ArrayList<>();
          Employee emp1 = new Employee(0.0,"D","Gaurav Chaudhary",0);
          Employee emp2 = new Employee(15_00000.0,"D","Gajodhar",12);
          Employee emp3 = new Employee(4_00000.0,"T","Puneet Saini",24);
          Employee emp4 = new Employee(8_00000.0,"D","Gajraj Patra",36);
          Employee emp5 = new Employee(12_00000,"D","Sumit Patidar",6);
          Employee emp6 = new Employee(11_00000,"D","Preeti Chauchan",48);
          Employee emp7 = new Employee(0.0,"D","Garima Garg",0);
          Employee emp8 = new Employee(7_00000,"T","Jatin R B",120);
          Employee emp9 = new Employee(8_00000,"D","Ravi L sahu",24);
        Employee emp10 = new Employee(0.0,"D","Brajesh B",0);

        employee_details1.add(emp1);
        employee_details1.add(emp2);
        employee_details1.add(emp3);
        employee_details1.add(emp4);
        employee_details1.add(emp5);
        employee_details2.add(emp6);
        employee_details2.add(emp7);
        employee_details2.add(emp8);
        employee_details2.add(emp9);
        employee_details2.add(emp10);

        ProcessEmployeeDetails ped1 = new ProcessEmployeeDetails(employee_details1);
        ProcessEmployeeDetails ped2 = new ProcessEmployeeDetails(employee_details2);

        System.out.println("Before Processing ..."+Thread.currentThread());

        for(int i = 0 ; i < employee_details1.size();i++) {
           System.out.println(employee_details1.get(i));
        }
        for(int i = 0 ; i < employee_details2.size();i++) {
           System.out.println(employee_details2.get(i));
        }

        ped1.start();
        ped2.start();


        System.out.println("After Processing ...");

        for(int i = 0 ; i < employee_details1.size();i++) {
           System.out.println(employee_details1.get(i));
        }
        for(int i = 0 ; i < employee_details2.size();i++) {
           System.out.println(employee_details2.get(i));
        }
     }
}
```

Output :-

```
Before Processing ...Thread[main,5,main]
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 0.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 0.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 0.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 0.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 0.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 0.0
After Processing ...
```

```
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 2025000.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 600000.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 1000000.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 1705000.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 1190000.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
```

Q2 – a)

Using Synchronized keyword in run()

```java
import java.util.ArrayList;

class Employee{
   /*Selected Employee have current salary, is their current salary in their previous company
    expected_salary, is their salary after the switch,
    fresher_salary_dev , is salary if Employee is fresher than what will be the salary of developer p.a.
    fresher_salary_test, is salary if Employee is fresher and selected for tester role p.a.
    type is the developer or tester role applied for.
    "D" : Developer role
    "T" : Tester role
    we have to compute expected_salary
   */
    double current_salary = 0.0,expected_salary = 0.0;
    int work_exp = 0;

    final double fresher_salary_dev = 10_00000, fresher_salary_test = 8_00000;
    String name,type;


   Employee(double current_salary, String type,String name,int work_exp){
      this.current_salary = current_salary;
      this.type = type;
      this.name = name;
      this.work_exp = work_exp;
   }

   @Override
   public String toString() {
      return "name :"+" "+this.name + ",Role :"+ this.type +",Current_salary :"+ this.current_salary
+",Expected_Salary: "+ this.expected_salary;
   }
}
 class ProcessEmployeeDetails extends Thread {
   /**
    *
    */
   ArrayList<Employee> employee_details = new ArrayList<>();
   int start,end,index;
   ProcessEmployeeDetails(ArrayList<Employee> employee_details){
      this.employee_details = employee_details;
   }

   @Override
   public synchronized void run() {
      // TODO Auto-generated method stub
         for(int i = 0 ;i < employee_details.size();i++){
            Employee fetched = employee_details.get(i);
            if(fetched.type.equals("D")) {
               if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
                  fetched.expected_salary = (75 * fetched.current_salary)/100 + fetched.current_salary;
               } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary ==
0.0) {
                  fetched.expected_salary = (55 * fetched.current_salary)/100 + fetched.current_salary;
               }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0)
{
                  fetched.expected_salary = (35 * fetched.current_salary)/100 + fetched.current_salary;
               } else {
                  fetched.expected_salary = fetched.fresher_salary_dev;
               }
            } else {
               if(fetched.work_exp >= 60 && fetched.expected_salary == 0.0) {
                  fetched.expected_salary = (70 * fetched.current_salary)/100 + fetched.current_salary;
               } else if(fetched.work_exp >= 24 && fetched.work_exp < 60 && fetched.expected_salary ==
0.0) {
                  fetched.expected_salary = (50 * fetched.current_salary)/100 + fetched.current_salary;
               }else if(fetched.work_exp >= 12 && fetched.work_exp < 24 && fetched.expected_salary == 0.0)
{
                  fetched.expected_salary = (30 * fetched.current_salary)/100 + fetched.current_salary;
               } else {
                  fetched.expected_salary = fetched.fresher_salary_test;
               }
            }
         }
      }
}
class Threading {
      public static void main(String... gaurav) throws Exception{
         //Creating employee list that got selected.
         ArrayList<Employee> employee_details1 = new ArrayList<>();
         ArrayList<Employee> employee_details2 = new ArrayList<>();
         Employee emp1 = new Employee(0.0,"D","Gaurav Chaudhary",0);
         Employee emp2 = new Employee(15_00000.0,"D","Gajodhar",12);
         Employee emp3 = new Employee(4_00000.0,"T","Puneet Saini",24);
         Employee emp4 = new Employee(8_00000.0,"D","Gajraj Patra",36);
         Employee emp5 = new Employee(12_00000,"D","Sumit Patidar",6);
         Employee emp6 = new Employee(11_00000,"D","Preeti Chauchan",48);
         Employee emp7 = new Employee(0.0,"D","Garima Garg",0);
         Employee emp8 = new Employee(7_00000,"T","Jatin R B",120);
         Employee emp9 = new Employee(8_00000,"D","Ravi L sahu",24);
         Employee emp10 = new Employee(0.0,"D","Brajesh B",0);
```

```java
        employee_details1.add(emp1);
        employee_details1.add(emp2);
        employee_details1.add(emp3);
        employee_details1.add(emp4);
        employee_details1.add(emp5);
        employee_details2.add(emp6);
        employee_details2.add(emp7);
        employee_details2.add(emp8);
        employee_details2.add(emp9);
        employee_details2.add(emp10);

        ProcessEmployeeDetails ped1 = new ProcessEmployeeDetails(employee_details1);
        ProcessEmployeeDetails ped2 = new ProcessEmployeeDetails(employee_details2);

        System.out.println("Before Processing ..."+Thread.currentThread());

        for(int i = 0 ; i < employee_details1.size();i++) {
           System.out.println(employee_details1.get(i));
        }
        for(int i = 0 ; i < employee_details2.size();i++) {
           System.out.println(employee_details2.get(i));
        }

        ped1.start();
        ped2.start();


        System.out.println("After Processing ...");

        for(int i = 0 ; i < employee_details1.size();i++) {
           System.out.println(employee_details1.get(i));
        }
        for(int i = 0 ; i < employee_details2.size();i++) {
           System.out.println(employee_details2.get(i));
        }
     }
}
```

Output :-

```
Before Processing ...Thread[main,5,main]
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 0.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 0.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 0.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 0.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 0.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 0.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 0.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 0.0
After Processing ...
name : Gaurav Chaudhary,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
name : Gajodhar,Role :D,Current_salary :1500000.0,Expected_Salary: 2025000.0
name : Puneet Saini,Role :T,Current_salary :400000.0,Expected_Salary: 600000.0
name : Gajraj Patra,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Sumit Patidar,Role :D,Current_salary :1200000.0,Expected_Salary: 1000000.0
name : Preeti Chauchan,Role :D,Current_salary :1100000.0,Expected_Salary: 1705000.0
name : Garima Garg,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
name : Jatin R B,Role :T,Current_salary :700000.0,Expected_Salary: 1190000.0
name : Ravi L sahu,Role :D,Current_salary :800000.0,Expected_Salary: 1240000.0
name : Brajesh B,Role :D,Current_salary :0.0,Expected_Salary: 1000000.0
```

Q2 b):

```java
import java.util.LinkedList;
import java.util.Queue;

class Producer implements Runnable{
   Queue<Integer> queue;
   int limit;
   static int start = 0;
   Producer(Queue<Integer> queue,int limit){
      this.queue = queue;
      this.limit = limit;
   }
   @Override
   public void run() {
      // TODO Auto-generated method stub
      System.out.println("Producer……… ");
      while(true){
         synchronized(queue){
            while(queue.size() == limit){
               try {
                  System.out.println("Queue is Full !");
                  queue.wait();
               } catch (InterruptedException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
               }
            }
            System.out.println("Producer thread produced "+start);
            queue.add(start);
            start += 1;
         // Notiy Consumer thread
            queue.notifyAll();
            if(start == 50){
               queue.add(50);
               System.out.println("Ending Producer thread");
               break;
            }
         }
      }

   }
}
class Consumer implements Runnable{
   Queue<Integer> queue;
   Consumer(Queue<Integer> queue){
      this.queue = queue;
   }

   @Override
   public void run() {
      // TODO Auto-generated method stub
      System.out.println("Consumer......... ");
      while(true){
         synchronized(queue){
            while(queue.isEmpty()){

            try {
               System.out.println("Queue is Empty !");
               queue.wait();

            } catch (InterruptedException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
               }

            }
         // Notify producer thread
            int rec = queue.poll();
            System.out.println("Consumer thread consumed "+rec);
            queue.notifyAll();
            if(rec == 50){
               System.out.println("Ending Consumer Thread");
               break;
            }
         }
      }

   }
}

class Main{
   public static void main(String[] args) {
      Queue<Integer> queue = new LinkedList<>();
      Producer produce = new Producer(queue, 10);
      Consumer consume = new Consumer(queue);

      Thread t1 = new Thread(produce);
      Thread t2 = new Thread(consume);

      t1.start();
      t2.start();

   }
}
```

Output :-

```
Consumer.....
Producer.....
Queue is Empty !
Producer thread produced 0
Producer thread produced 1
Consumer thread consumed 0
Consumer thread consumed 1
Producer thread produced 2
Consumer thread consumed 2
Queue is Empty !
Producer thread produced 3
Producer thread produced 4
Producer thread produced 5
Producer thread produced 6
Producer thread produced 7
Producer thread produced 8
Producer thread produced 9
Producer thread produced 10
Producer thread produced 11
Producer thread produced 12
Queue is Full !
Consumer thread consumed 3
Consumer thread consumed 4
Consumer thread consumed 5
Producer thread produced 13
Producer thread produced 14
Producer thread produced 15
Queue is Full !
Consumer thread consumed 6
Consumer thread consumed 7
Consumer thread consumed 8
Consumer thread consumed 9
Consumer thread consumed 10
Producer thread produced 16
Producer thread produced 17
Producer thread produced 18
Producer thread produced 19
Producer thread produced 20
Queue is Full !
Consumer thread consumed 11
Consumer thread consumed 12
Consumer thread consumed 13
Consumer thread consumed 14
Consumer thread consumed 15
Consumer thread consumed 16
Consumer thread consumed 17
Consumer thread consumed 18
Consumer thread consumed 19
Consumer thread consumed 20
Queue is Empty !
Producer thread produced 21
Producer thread produced 22
Producer thread produced 23
Producer thread produced 24
Consumer thread consumed 21
Consumer thread consumed 22
Consumer thread consumed 23
Producer thread produced 25
Producer thread produced 26
Producer thread produced 27
Producer thread produced 28
Producer thread produced 29
Producer thread produced 30
Producer thread produced 31
Producer thread produced 32
Producer thread produced 33
Queue is Full !
Consumer thread consumed 24
Consumer thread consumed 25
Producer thread produced 34
Producer thread produced 35
Queue is Full !
Consumer thread consumed 26
Consumer thread consumed 27
Consumer thread consumed 28
Consumer thread consumed 29
Consumer thread consumed 30
Producer thread produced 36
Producer thread produced 37
Consumer thread consumed 31
Consumer thread consumed 32
Consumer thread consumed 33
Consumer thread consumed 34
Consumer thread consumed 35
Consumer thread consumed 36
Consumer thread consumed 37
Producer thread produced 38
Producer thread produced 39
Producer thread produced 40
Producer thread produced 41
Producer thread produced 42
Producer thread produced 43
Producer thread produced 44
Producer thread produced 45
Producer thread produced 46
Producer thread produced 47
Queue is Full !
Consumer thread consumed 38
Consumer thread consumed 39
Consumer thread consumed 40
Consumer thread consumed 41
Consumer thread consumed 42
Consumer thread consumed 43
Consumer thread consumed 44
Consumer thread consumed 45
Consumer thread consumed 46
Consumer thread consumed 47
Queue is Empty !
Producer thread produced 48
Producer thread produced 49
Ending Producer thread
Consumer thread consumed 48
Consumer thread consumed 49
Consumer thread consumed 50
Ending Consumer Thread
```

Q2 C):

```java
import java.util.concurrent.atomic.AtomicInteger;

 class AtomicClasses {
  public static void main(String args[])
  {

     // Initially value as 0
     AtomicInteger val
        = new AtomicInteger(0);

     // Decreases and gets
     // the previous value
     int res = val.getAndIncrement();

     System.out.println("Previous value: "+ res);

     // Prints the updated value
     System.out.println("Current value: "+ val);

     // Increment and get
     res = val.incrementAndGet();

     // Prints the updated value
     System.out.println("Current value: " + res);

     //Decrement and get
     res = val.decrementAndGet();

     // Prints the updated value
     System.out.println("Current value: " + res);

     boolean res2 = val.compareAndSet(1, 5);

     // Prints the updated value
     System.out.println("Current value: " + val +" "+res2);
  }
}
```

output ->

```
Previous value: 0
Current value: 1
Current value: 2
Current value: 1
Current value: 5 true
```

Problem Statement 2:-

1:-

```
class ThreadDemo extends Thread{
  ThreadDemo(String ThreadName){
     super(ThreadName);
  }
  public void run(){
        System.out.println("ThreadDemo Running……. ");
        System.out.println("ThreadDemo name is :"+Thread.currentThread().getName());
        System.out.println("ThreadDemo ID :"+Thread.currentThread().getId());
        System.out.println("ThreadDemo Priority :"+Thread.currentThread().getPriority());
        System.out.println("ThreadDemo Phase :"+Thread.currentThread().getStackTrace().toString());
    }
}
class Main{
    public static void main(String[] args) {
      ThreadDemo t = new ThreadDemo("ThreadDemoClass_Thread");
      t.setPriority(10);
      t.start();
      System.out.println("Main Thread Priority :"+Thread.currentThread().getPriority());
      System.out.println("Main Thread Name :"+Thread.currentThread().getName());
    }
}
```

Output :-

```
Main Thread Priority :5
ThreadDemo Running....
ThreadDemo name is :ThreadDemoClass_Thread
Main Thread Name :main
ThreadDemo ID :14
ThreadDemo Priority :10
ThreadDemo Phase :[Ljava.lang.StackTraceElement;@171e54f1
```

1 :- Using Runnable Interaface

```
class ThreadDemo implements Runnable{

    public void run(){
        System.out.println("ThreadDemo Running....... ");
        System.out.println("ThreadDemo name is :"+Thread.currentThread().getName());
        System.out.println("ThreadDemo ID :"+Thread.currentThread().getId());
        System.out.println("ThreadDemo Priority :"+Thread.currentThread().getPriority());
        System.out.println("ThreadDemo Phase :"+Thread.currentThread().getStackTrace().toString());
    }
}
class Main{
    public static void main(String[] args) {
      ThreadDemo th = new ThreadDemo();
      Thread t = new Thread(th);
      t.setPriority(10);
      t.setName("ThreadDemoClass_Thread");
      t.start();
      System.out.println("Main Thread Priority :"+Thread.currentThread().getPriority());
      System.out.println("Main Thread Name :"+Thread.currentThread().getName());
    }
}
```

Output ->

```
ThreadDemo Running....
ThreadDemo name is :ThreadDemoClass_Thread
Main Thread Priority :5
ThreadDemo ID :14
Main Thread Name :main
ThreadDemo Priority :10
ThreadDemo Phase :[Ljava.lang.StackTraceElement;@d12e302
```

Q2

```
class PrintingNumber implements Runnable{
   int start,end;

   PrintingNumber(int start,int end){
      this.start = start;
      this.end = end;
   }
   public void run(){
         for(int i = start; i <= end; i++){
            System.out.println(Thread.currentThread().getName()+" Printed "+i);
         }
   }
}
class Main{
   public static void main(String[] args) {
      PrintingNumber pn1 = new PrintingNumber(101, 200);
      PrintingNumber pn2 = new PrintingNumber(201, 300);
      PrintingNumber pn3 = new PrintingNumber(301, 400);

      // creating thread class

      Thread t1 = new Thread(pn1);
      Thread t2 = new Thread(pn2);
      Thread t3 = new Thread(pn3);

      // giving name to each thread

      t1.setName("101-200 Thread");
      t2.setName("201-300 Thread");
      t3.setName("301-400 Thread");

      // starting thread
      t1.start();
      t2.start();
      t3.start();

   }
}
```

Output : -

```
301-400 Thread Printed 301
```

```
301-400 Thread Printed 302
201-300 Thread Printed 201
201-300 Thread Printed 202
201-300 Thread Printed 203
201-300 Thread Printed 204
101-200 Thread Printed 101
101-200 Thread Printed 102
101-200 Thread Printed 103
101-200 Thread Printed 104
201-300 Thread Printed 205
201-300 Thread Printed 206
301-400 Thread Printed 303
301-400 Thread Printed 304
201-300 Thread Printed 207
201-300 Thread Printed 208
201-300 Thread Printed 209
201-300 Thread Printed 210
201-300 Thread Printed 211
201-300 Thread Printed 212
101-200 Thread Printed 105
201-300 Thread Printed 213
201-300 Thread Printed 214
201-300 Thread Printed 215
201-300 Thread Printed 216
201-300 Thread Printed 217
301-400 Thread Printed 305
201-300 Thread Printed 218
201-300 Thread Printed 219
101-200 Thread Printed 106
201-300 Thread Printed 220
201-300 Thread Printed 221
301-400 Thread Printed 306
201-300 Thread Printed 222
201-300 Thread Printed 223
101-200 Thread Printed 107
101-200 Thread Printed 108
101-200 Thread Printed 109
101-200 Thread Printed 110
101-200 Thread Printed 111
101-200 Thread Printed 112
201-300 Thread Printed 224
201-300 Thread Printed 225
101-200 Thread Printed 113
301-400 Thread Printed 307
301-400 Thread Printed 308
301-400 Thread Printed 309
101-200 Thread Printed 114
101-200 Thread Printed 115
101-200 Thread Printed 116
101-200 Thread Printed 117
101-200 Thread Printed 118
101-200 Thread Printed 119
```

```
301-400 Thread Printed 310
201-300 Thread Printed 226
201-300 Thread Printed 227
201-300 Thread Printed 228
201-300 Thread Printed 229
201-300 Thread Printed 230
201-300 Thread Printed 231
201-300 Thread Printed 232
201-300 Thread Printed 233
101-200 Thread Printed 120
301-400 Thread Printed 311
101-200 Thread Printed 121
201-300 Thread Printed 234
201-300 Thread Printed 235
101-200 Thread Printed 122
301-400 Thread Printed 312
301-400 Thread Printed 313
301-400 Thread Printed 314
301-400 Thread Printed 315
301-400 Thread Printed 316
301-400 Thread Printed 317
301-400 Thread Printed 318
301-400 Thread Printed 319
301-400 Thread Printed 320
301-400 Thread Printed 321
301-400 Thread Printed 322
301-400 Thread Printed 323
101-200 Thread Printed 123
201-300 Thread Printed 236
201-300 Thread Printed 237
101-200 Thread Printed 124
101-200 Thread Printed 125
101-200 Thread Printed 126
301-400 Thread Printed 324
301-400 Thread Printed 325
101-200 Thread Printed 127
101-200 Thread Printed 128
201-300 Thread Printed 238
201-300 Thread Printed 239
201-300 Thread Printed 240
201-300 Thread Printed 241
101-200 Thread Printed 129
101-200 Thread Printed 130
301-400 Thread Printed 326
301-400 Thread Printed 327
101-200 Thread Printed 131
201-300 Thread Printed 242
301-400 Thread Printed 328
301-400 Thread Printed 329
301-400 Thread Printed 330
301-400 Thread Printed 331
101-200 Thread Printed 132
```

```
101-200 Thread Printed 133
301-400 Thread Printed 332
301-400 Thread Printed 333
301-400 Thread Printed 334
301-400 Thread Printed 335
201-300 Thread Printed 243
301-400 Thread Printed 336
101-200 Thread Printed 134
101-200 Thread Printed 135
101-200 Thread Printed 136
101-200 Thread Printed 137
101-200 Thread Printed 138
101-200 Thread Printed 139
301-400 Thread Printed 337
301-400 Thread Printed 338
301-400 Thread Printed 339
201-300 Thread Printed 244
201-300 Thread Printed 245
201-300 Thread Printed 246
301-400 Thread Printed 340
101-200 Thread Printed 140
101-200 Thread Printed 141
301-400 Thread Printed 341
301-400 Thread Printed 342
301-400 Thread Printed 343
201-300 Thread Printed 247
301-400 Thread Printed 344
101-200 Thread Printed 142
101-200 Thread Printed 143
301-400 Thread Printed 345
301-400 Thread Printed 346
301-400 Thread Printed 347
201-300 Thread Printed 248
201-300 Thread Printed 249
201-300 Thread Printed 250
101-200 Thread Printed 144
101-200 Thread Printed 145
201-300 Thread Printed 251
201-300 Thread Printed 252
201-300 Thread Printed 253
201-300 Thread Printed 254
201-300 Thread Printed 255
201-300 Thread Printed 256
201-300 Thread Printed 257
201-300 Thread Printed 258
201-300 Thread Printed 259
201-300 Thread Printed 260
201-300 Thread Printed 261
201-300 Thread Printed 262
201-300 Thread Printed 263
201-300 Thread Printed 264
201-300 Thread Printed 265
```

```
201-300 Thread Printed 266
301-400 Thread Printed 348
201-300 Thread Printed 267
201-300 Thread Printed 268
201-300 Thread Printed 269
201-300 Thread Printed 270
201-300 Thread Printed 271
201-300 Thread Printed 272
201-300 Thread Printed 273
201-300 Thread Printed 274
101-200 Thread Printed 146
101-200 Thread Printed 147
101-200 Thread Printed 148
101-200 Thread Printed 149
201-300 Thread Printed 275
301-400 Thread Printed 349
301-400 Thread Printed 350
301-400 Thread Printed 351
301-400 Thread Printed 352
201-300 Thread Printed 276
101-200 Thread Printed 150
101-200 Thread Printed 151
101-200 Thread Printed 152
101-200 Thread Printed 153
101-200 Thread Printed 154
101-200 Thread Printed 155
101-200 Thread Printed 156
301-400 Thread Printed 353
301-400 Thread Printed 354
301-400 Thread Printed 355
301-400 Thread Printed 356
101-200 Thread Printed 157
201-300 Thread Printed 277
201-300 Thread Printed 278
101-200 Thread Printed 158
301-400 Thread Printed 357
101-200 Thread Printed 159
201-300 Thread Printed 279
201-300 Thread Printed 280
101-200 Thread Printed 160
101-200 Thread Printed 161
101-200 Thread Printed 162
301-400 Thread Printed 358
301-400 Thread Printed 359
301-400 Thread Printed 360
101-200 Thread Printed 163
101-200 Thread Printed 164
201-300 Thread Printed 281
101-200 Thread Printed 165
301-400 Thread Printed 361
301-400 Thread Printed 362
301-400 Thread Printed 363
```

```
201-300 Thread Printed 282
201-300 Thread Printed 283
201-300 Thread Printed 284
201-300 Thread Printed 285
201-300 Thread Printed 286
301-400 Thread Printed 364
301-400 Thread Printed 365
101-200 Thread Printed 166
201-300 Thread Printed 287
201-300 Thread Printed 288
201-300 Thread Printed 289
201-300 Thread Printed 290
201-300 Thread Printed 291
201-300 Thread Printed 292
201-300 Thread Printed 293
201-300 Thread Printed 294
101-200 Thread Printed 167
201-300 Thread Printed 295
201-300 Thread Printed 296
301-400 Thread Printed 366
201-300 Thread Printed 297
101-200 Thread Printed 168
201-300 Thread Printed 298
301-400 Thread Printed 367
301-400 Thread Printed 368
301-400 Thread Printed 369
201-300 Thread Printed 299
201-300 Thread Printed 300
101-200 Thread Printed 169
101-200 Thread Printed 170
301-400 Thread Printed 370
301-400 Thread Printed 371
301-400 Thread Printed 372
101-200 Thread Printed 171
101-200 Thread Printed 172
301-400 Thread Printed 373
101-200 Thread Printed 173
101-200 Thread Printed 174
101-200 Thread Printed 175
101-200 Thread Printed 176
101-200 Thread Printed 177
101-200 Thread Printed 178
101-200 Thread Printed 179
101-200 Thread Printed 180
101-200 Thread Printed 181
301-400 Thread Printed 374
301-400 Thread Printed 375
301-400 Thread Printed 376
101-200 Thread Printed 182
101-200 Thread Printed 183
101-200 Thread Printed 184
101-200 Thread Printed 185
```

```
301-400 Thread Printed 377
101-200 Thread Printed 186
101-200 Thread Printed 187
101-200 Thread Printed 188
101-200 Thread Printed 189
301-400 Thread Printed 378
301-400 Thread Printed 379
301-400 Thread Printed 380
101-200 Thread Printed 190
101-200 Thread Printed 191
301-400 Thread Printed 381
301-400 Thread Printed 382
301-400 Thread Printed 383
101-200 Thread Printed 192
101-200 Thread Printed 193
101-200 Thread Printed 194
101-200 Thread Printed 195
301-400 Thread Printed 384
301-400 Thread Printed 385
301-400 Thread Printed 386
301-400 Thread Printed 387
301-400 Thread Printed 388
301-400 Thread Printed 389
301-400 Thread Printed 390
301-400 Thread Printed 391
301-400 Thread Printed 392
301-400 Thread Printed 393
101-200 Thread Printed 196
101-200 Thread Printed 197
301-400 Thread Printed 394
301-400 Thread Printed 395
301-400 Thread Printed 396
301-400 Thread Printed 397
301-400 Thread Printed 398
101-200 Thread Printed 198
101-200 Thread Printed 199
301-400 Thread Printed 399
101-200 Thread Printed 200
301-400 Thread Printed 400
```

Conclusion :- Since the threads are not synchronized, we can say that they are running randomly or simultaneously.