a)

Problem Description:

- The current problem is that there are currently discrepancies in overtime calculations despite precise record-keeping, there are multiple errors in overtime pay that have surfaced.

- Employees in the company have reported numerous times that their hard-earned overtime hours were not accurately reflected in their paychecks. This has led to heated discussions during break times and after-shift gatherings.

- A meeting was held to find meaningful solutions on how their overtime for different shifts can be calculated accurately and provide transparency on retirement plan contributions.

b)

Research Integration:

- The data that will be required include the number of hours an employee has worked, the shift they worked, the hourly pay rate, the regular pay and the overtime pay. Based on the data, an employee's netpay will be calculated by adding their regular pay and their overtime pay.

- The program will calculate regular pay based on the standard hourly rates an employee works in a week. For example if an employee works for 40 hours and works under shift 1, the number of hours of hours they worked will be multiplied by R50 which is the hourly pay rate for shift 1.

- The program will have to calculate the overtime pay based on the shift an employee worked and work hours that exceed 40 hours a week. So if an employee worked for 50 hours in one week, overtime will be calculated by multiplying the difference between the hours they worked and 40 hours and multiplying the result by R75 which we get from overtime rate which is 1.5 times the standard rate.

- An employee has the option to opt into the retirement plan and should they decide to opt into it then 5% of their gross pay will be deducted and put towards the retirement plan.

c)

Viability Evaluation:

- Due to the nature of the program, it is feasible to develop a program to address the current problem since there is currently no system in place that accurately calculates overtime pay for an employee. This leads to discrepancies in the paychecks that the employees receive which note that there are inaccuracies in the paycheck.

- The reason for it being feasible is that overtime pay has been calculated manually which has led to overtime not being accurately calculated and has led to the dissatisfaction of multiple employees since their pay is not being accurately calculated. This discourages employees from wanting to work overtime since they will know that their overtime pay will not be calculated and paid correctly.

- The current problem can be solved with current technology within the constraints. The program does not require a lot of resources and is very easy for management to use. It does not require a lot of resources to create, and it will end up being extremely useful since it will accurately calculate employee's overtime pay.

- The technology that will be used is NetBeans Apache which is easy to use and understand since it is not complex and difficult to use, even for people who have never used it before.

d)

Evaluating the economic viability:

- The benefits of the program do outweigh the costs of development. This is because should the problem go unsolved, it will harm the business due to unhappy employees which in turn affects the business quality of work and products. This also helps to

- The main purpose of the program is to solve the problem that the business is currently facing as it can accurately calculate the bonus of employees and accurately display the amount they earned on their payslips without any discrepancies.

- The reason for the benefits outweighing the costs of development is that should they decide not to use the program then they will have to manually

calculate the employee's bonuses which leads to discrepancies which is how the problem initially started.

- The overall cost of the project is estimated between R36 500 and R54 000.

e)

Solution Selection:

- The most suitable solution for the problem is to create a program that will calculate an employee's overtime pay accurately without there being any discrepancies.

- The reason, why this solution is most suitable, is because should the discrepancies in the payroll continue, a lot of employees will decide to leave since they will not be receiving the money they are meant to receive. The company would have to spend R10 000 for the recruiting process per employee and that includes having to do orientation for the employee which is time-consuming.

- The solution 1 offers a cheaper option of having to spend only R300 an hour on one developer to work on the program as this will help save time and money instead of having to use that money for inducting new employees.

- The solution 1 has a high return on investment since it will be used to now accurately calculate an employee's overtime pay without there being any discrepancies which then leads to high morale in employees increases the quality of work produced by the company and leads to more customers being satisfied with the services they receive.

- The solution chosen takes less time to code since everything is split across 2 classes instead of everything being in one class and uses unnecessary if statements.

- These were the 2 solutions in comparison:

    - **Solution 1:**
        - Uses 2 classes which are the main class and the employee class.
        - Uses encapsulation within the employee class to be able to get values.

- Takes less time to create since it does not use a lot of code.
- Since it takes less time to create, it costs less.
- The code is readable and easy to understand with comments.
- Since only one week will be spent on, a total of R16 800 will be spent.

```java
package andrew_payroll;

import java.util.*;
import java.io.*;

public class Andrew_Payroll {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        ArrayList<Employee> employees = new ArrayList<>();

        try {
            RandomAccessFile file = new RandomAccessFile("hours_worked.txt", "rw");

            System.out.println("Please enter how many hours you worked:");
            int hoursWorked = sc.nextInt();
            if (hoursWorked < 0) {
                System.out.println("Hours worked cannot be negative");
                return;
            }

            System.out.println("Please enter your shift number:");
            int shiftNumber = sc.nextInt();
            if (shiftNumber < 1 || shiftNumber > 3) {
                System.out.println("Invalid shift number. Please enter 1, 2, or 3.");
                return;
            }

            boolean retirementPlan = false;

            if (shiftNumber == 2 || shiftNumber == 3) {
                System.out.println("Do you wish to participate in the retirement plan? (yes/no)");
                String choice = sc.next();
                if ("yes".equalsIgnoreCase(choice)) {
                    retirementPlan = true;
                }
            }

            Employee employee = new Employee(hoursWorked, shiftNumber, retirementPlan);
            employees.add(employee);
            employee.calculatePay();

            displayEmployeeDetails(employee);

            file.seek(file.length());
            file.writeBytes(hoursWorked + "\n");

            file.close();
        } catch (IOException e) {
            System.out.println("An error occurred while handling the file.");
        }
    }

```

```
    private static void displayEmployeeDetails(Employee employee) {
        System.out.println("Employee Details:");
        System.out.println("Shift Number: " + employee.getShiftNumber());
        System.out.println("Hours Worked: " + employee.getHoursWorked());
        System.out.println("Regular Pay: " + "R" + employee.getRegularPay());
        System.out.println("Overtime Pay: " + "R" + employee.getOvertimePay());
        System.out.println("Total Pay: " + "R" + employee.getTotalPay());
        System.out.println("Retirement Deduction: " + "R" + employee.getRetirementDeduction());
        System.out.println("Net Pay: " + "R" + employee.getNetPay());
    }
}
```

```
1   package andrew_payroll;
2
3   import java.io.*;
4
5   public class Employee {
6
7       private int hoursWorked;
8       private int shiftNumber;
9       private boolean hasRetirementPlan;
10      private double regularPay;
11      private double overtimePay;
12      private double totalPay;
13      private double retirementDeduction;
14      private double netPay;
15      private boolean validShift;
16
17      public Employee(int hoursWorked, int shiftNumber, boolean hasRetirementPlan) {
18          this.hoursWorked = hoursWorked;
19          this.shiftNumber = shiftNumber;
20          this.hasRetirementPlan = hasRetirementPlan;
21          this.validShift = shiftNumber >= 1 && shiftNumber <= 3;
22      }
23
24      public void calculatePay() {
25          if (hoursWorked < 0) {
26              throw new IllegalArgumentException("Hours worked cannot be negative.");
27          }
28
29          if (!validShift) {
30              regularPay = 0;
31              overtimePay = 0;
32              totalPay = 0;
33              retirementDeduction = 0;
34              netPay = 0;
35              return;
36          }
37
38          double hourlyPayRate;
            switch (shiftNumber) {
40              case 1:
41                  hourlyPayRate = 50;
42                  break;
43              case 2:
44                  hourlyPayRate = 70;
45                  break;
```

```
46            case 3:
47                hourlyPayRate = 90;
48                break;
49            default:
50                hourlyPayRate = 0;
51                break;
52        }
53
54        if (hoursWorked > 40) {
55            regularPay = 40 * hourlyPayRate;
56            overtimePay = (hoursWorked - 40) * hourlyPayRate * 1.5;
57        } else {
58            regularPay = hoursWorked * hourlyPayRate;
59            overtimePay = 0;
60        }
61
62        totalPay = regularPay + overtimePay;
63
64        if (shiftNumber != 1 && hasRetirementPlan) {
65            retirementDeduction = 0.05 * totalPay;
66        } else {
67            retirementDeduction = 0;
68        }
69
70        netPay = totalPay - retirementDeduction;
71    }
72
73    public int getHoursWorked() {
74        return hoursWorked;
75    }
76
77    public int getShiftNumber() {
78        return shiftNumber;
79    }
80
81    public boolean hasRetirementPlan() {
82        return hasRetirementPlan;
83    }
84
```

```java
    public double getRegularPay() {
        return regularPay;
    }

    public double getOvertimePay() {
        return overtimePay;
    }

    public double getTotalPay() {
        return totalPay;
    }

    public double getRetirementDeduction() {
        return retirementDeduction;
    }

    public double getNetPay() {
        return netPay;
    }

    public boolean isValidShift() {
        return validShift;
    }
```

```java
    public static void appendHoursToFile(int hoursWorked) throws IOException {
        FileWriter fw = new FileWriter("hours_worked.txt", true);
        fw.write(hoursWorked + "\n");
        fw.close();
    }

    public static void displayFileContents(String filename) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader(filename));
        String line;
        while ((line = br.readLine()) != null) {
            System.out.println(line);
        }
        br.close();
    }
}
```

- **Solution 2:**
  - Solution 2 has unreadable code since everything is all in one class.
  - Uses nested ifs for all the problems which then makes the code unreadable.
  - Takes more time to implement since more time will be spent on it due to there being multiple lines of code.

- It will cost more to implement this solution because it will take more than a week when that Is the time frame and the estimated cost ends up being R56 000.

```java

package andrew_payroll;
import java.util.*;


public class Andrew_Payroll {

  public static void main(String[] args) {

    Scanner sc = new Scanner (System.in);
    Scanner cs = new Scanner (System.in);

    System.out.println("Please enter how many hours you worked");
    int hoursworked = sc.nextInt();

    System.out.println("Please enter your shift number: ");
    int shiftnum = sc.nextInt();

    if (shiftnum == 1) {

      double hourlypayrate = 50;
      double regularpay = hoursworked * hourlypayrate;
      if (hoursworked>40){
        regularpay = 40 * hourlypayrate;
      }
      int overtime = hoursworked - 40;
      double overtimepay = overtime * 75;
      double reovt = regularpay + overtimepay;

      System.out.println("Hours worked: " + hoursworked);
      System.out.println("Shift number: " + shiftnum);
      System.out.println("Hourly pay rate: " + "R" + hourlypayrate  + " per hour");
      System.out.println("Regular pay: " + "R" + regularpay );
      System.out.println("Overtime pay: " + "R" + overtimepay);
      System.out.println("Total of regular and overtime pay: " + "R" + reovt);
      System.out.println("Net pay: " + "R" + reovt);

    }

    if (shiftnum == 2) {

      double hourlypayrate = 70;
      double regularpay = hoursworked * hourlypayrate;

      if (hoursworked>40){
        regularpay = 40 * hourlypayrate;
      }

      int overtime = hoursworked - 40;
      double overtimepay = overtime * 105;
      double reovt = regularpay + overtimepay;
      double redeuct = 0.05 * reovt;
      double netpay = reovt - redeuct;
```

```
        System.out.println("Hours worked: " + hoursworked);
        System.out.println("Shift number: " + shiftnum);
        System.out.println("Hourly pay rate: " + "R" + hourlypayrate  + " per hour");
        System.out.println("Regular pay: " + "R" + regularpay );
        System.out.println("Overtime pay: " + "R" + overtimepay);

        System.out.println("Do you wish to participate in the retirement plan? (yes/no)");
        String choice = cs.nextLine();

        if ("yes".equals(choice)) {

           System.out.println("Retirement deduction: " + "R"  + redeuct);

        }

        System.out.println("Total of regular and overtime pay: " + "R" + reovt);
        System.out.println("Net pay: " + "R" + netpay);

    }

```

```
            if (shiftnum == 3) {

          double hourlypayrate = 90;
          double regularpay = hoursworked * hourlypayrate;
          if (hoursworked > 40){
             regularpay = 40 * hourlypayrate;
          }
          int overtime = hoursworked - 40;
          double overtimepay = overtime * 135;
          double reovt = regularpay + overtimepay;
          double redeuct = 0.05 * reovt;
          double netpay = reovt - redeuct;

          System.out.println("Hours worked: " + hoursworked);
          System.out.println("Shift number: " + shiftnum);
          System.out.println("Hourly pay rate: " + "R" + hourlypayrate  + " per hour");
          System.out.println("Regular pay: " + "R" + regularpay );
          System.out.println("Overtime pay: " + "R" + overtimepay);

```

```
        System.out.println("Do you wish to participate in the retirement plan? (yes/no)");
        String choice = cs.nextLine();

        if ("yes".equals(choice)) {

           System.out.println("Retirement deduction: " + "R"  + redeuct);

        }
        System.out.println("Total of regular and overtime pay: " + "R" + reovt);
        System.out.println("Net pay: " + "R" + netpay);


    }
```