Computer Program Solutions

QUESTION 4

Ethan Van Rensburg – System Development
Learnership Candidate

18 – 23 June 2024

Contents

a) Test Assessment:

Objectives

Determine the Range of Hours:

In the program's requirements, a work week is needed to be analyzed for both regular time and overtime. The regular/base working hours are 40, which we can identify as our minimum. There can also be additional hours, this being overtime, which exceed our marked base of 40 hours.

Determine the Maximum number of Users:

Due to UrbanFurn's factory-based operations and the company being based in industrial workings, the workforce on average ranges between operations workers, accountants, managers and supervisors. The average amount estimated by UrbanFurn for operational procedures and payroll delivery was about 400 users. Therefore, the system should be noted for delivery for 400 end users within UrbanFurn's operations and their software equipment.

Determine the validity of Calculations:

The program's main goal is to provide accurate and suitable calculations for regular pay and overtime pay towards the factory workers of UrbanFurn and their weekly salary. Therefore, the manual and unit tests will search for these suitable results and will correct any issues if they are present.

Determine the relationship between Shifts:

All shifts follow a similar setup of calculations and input requirements to display results. With this in mind, the program must be tested on the relationship between these sections of the main code to discover their underlying links and cooperation.

Determine the connections between methods:

The program is comprised of multiple methods, the main method includes the calculation of the different shifts, but alongside it are other methods. The remaining four methods help to calculate retirement deductions for shifts 2 and 3, based on regular pay and overtime pay. These methods must be tested in their validity and sharing of variables and calculations to determine if they are suitable for final outputs and displays.

Test Strategy Development:

Cycle 1:

1. When run, the program displays the 3 shifts of UrbanFurn in a numbered list. Alongside the display, the system also allows the user to enter the shift they operate in.
2. After the number is entered, the user will be prompted to enter their hours of work.
3. After the needed values are inputted, the user will be given their results and calculations.
4. If the user had chosen shift 2 or 3, they will be presented with the retirement plan prompt.
5. Choosing no will not add the deductions to the net pay.
6. Choosing yes will add the deductions to the net pay.
7. The same can be done within shift 3.
8. Alongside the regular pay, there is also the overtime calculations for shift 1.
9. Alongside the regular pay, there is also the overtime calculations for shift 2.
10. Alongside the regular pay, there is also the overtime calculations for shift 3.
11. If the user inputs an incorrect shift number, the system will reject their action and they will need to re-run the program.
12. If the user inputs an incorrect number of work hours, they will then be rejected and will need to re-run the program.
13. All hours inputted are processed and saved into a text file called Hours.

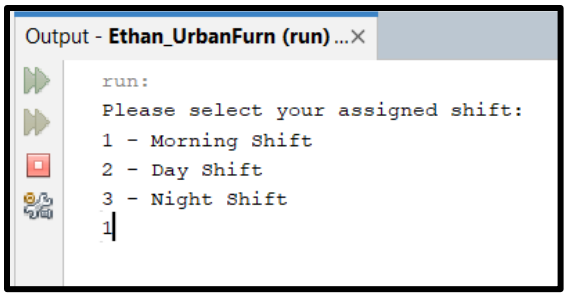
Cycle 2:

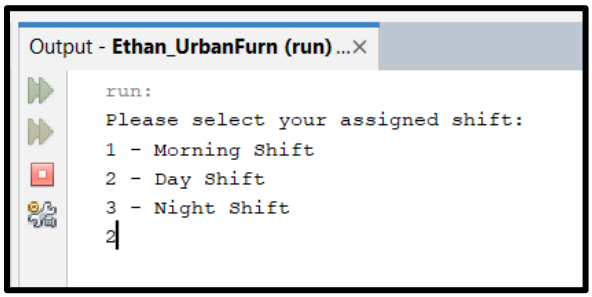
The second cycle will involve the implementation of unit testing for the program's main functions and methods. The test cases will test the minimum and maximum work hours allowed by the payroll system. The main case will test the main method for the 3 shifts and their calculations, while the other 4 test cases will test the retire methods for regular and overtime pay of shifts 2 and 3.

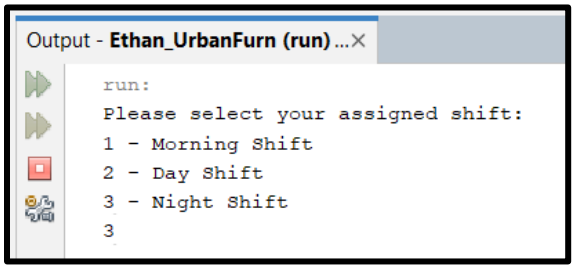
b) Test Result Recording:

Cycle 1:

1:







2:

```
Output - Ethan_UrbanFurn (run) ...
run:
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
1
How many hours do you work in a week?
40
```

```
Output - Ethan_UrbanFurn (run) ...
run:
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
2
How many hours do you work in a week?
40
```

```
Output - Ethan_UrbanFurn (run) ...
run:
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
3
How many hours do you work in a week?
40
```

3:

```
Payroll:
----------
Hours Worked: 40
Shift: Morning
Hourly Pay Rate: R50.0
Regular Pay: R2000
Overtime Pay: R0.00
Total of Regular and Overtime: R2000
Retirment Deduction: R0.00
Net Pay: R2000
BUILD SUCCESSFUL (total time: 7 seconds)
```

4:

```
Do you wish to sign up for the retirement plan? (yes/no)
no
```

```
Do you wish to sign up for the retirement plan? (yes/no)
yes
```

5:

```
Payroll:
----------
Hours Worked: 40
Shift: Day
Hourly Pay Rate: R70.00
Regular Pay: R2800
Overtime Pay: R0.00
Total of Regular and Overtime: R2800
Retirment Deduction: R0.00
Net Pay: R2800
BUILD SUCCESSFUL (total time: 3 minutes 56 seconds)
```

6:

```
Payroll:
----------
Hours Worked: 40
Shift: Day
Hourly Pay Rate: R70.0
Regular Pay: R2800
Overtime Pay: R0.00
Total of Regular and Overtime: R2800
Retirment Deduction: R140.0
Net Pay: R2660.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

7:

```
Payroll:
----------
Hours Worked: 40
Shift: Night
Hourly Pay Rate: R90.00
Regular Pay: R3600
Overtime Pay: R0.00
Total of Regular and Overtime: R3600
Retirment Deduction: R0.00
Net Pay: R3600
BUILD SUCCESSFUL (total time: 6 seconds)
```

```
Payroll:
----------
Hours Worked: 40
Shift: Night
Hourly Pay Rate: R90.0
Regular Pay: R3600
Overtime Pay: R0.00
Total of Regular and Overtime: R3600
Retirment Deduction: R180.0
Net Pay: R3420.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

8:

```
Payroll:
----------
Hours Worked: 50
Shift: Morning
Hourly Pay Rate: R50.0
Regular Pay: R2000
Overtime Pay: R750
Total of Regular and Overtime: R2750
Retirment Deduction: R0.00
Net Pay: R2750
BUILD SUCCESSFUL (total time: 8 seconds)
```

9:

```
Payroll:
----------
Hours Worked: 50
Shift: Day
Hourly Pay Rate: R70.00
Regular Pay: R2800
Overtime Pay: R1050
Total of Regular and Overtime: R3850
Retirment Deduction: R0.00
Net Pay: R3850
BUILD SUCCESSFUL (total time: 7 seconds)
```

```
Payroll:
----------
Hours Worked: 50
Shift: Day
Hourly Pay Rate: R70.0
Regular Pay: R2800
Overtime Pay: R1050
Total of Regular and Overtime: R3850
Retirment Deduction: R192.0
Net Pay: R3658.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

10:

```
Payroll:
----------
Hours Worked: 50
Shift: Night
Hourly Pay Rate: R90.00
Regular Pay: R3600
Overtime Pay: R1350
Total of Regular and Overtime: R4950
Retirment Deduction: R0.00
Net Pay: R4950
BUILD SUCCESSFUL (total time: 8 seconds)
```

```
Payroll:
----------
Hours Worked: 50
Shift: Night
Hourly Pay Rate: R90.0
Regular Pay: R3600
Overtime Pay: R1350
Total of Regular and Overtime: R4950
Retirment Deduction: R247.0
Net Pay: R4703.0
BUILD SUCCESSFUL (total time: 6 seconds)
```

11:

```
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
5
Invalid input
BUILD SUCCESSFUL (total time: 3 seconds)
```

12:

```
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
2
How many hours do you work in a week?
0
Invalid input
BUILD SUCCESSFUL (total time: 4 seconds)
```

```
Please select your assigned shift:
1 - Morning Shift
2 - Day Shift
3 - Night Shift
3
How many hours do you work in a week?
-67
Invalid input
BUILD SUCCESSFUL (total time: 6 seconds)
```

13:

```
Hours Worked: 45
Hours Worked: 70
Hours Worked: 40
Hours Worked: 70
Hours Worked: 40
Hours Worked: 70
Hours Worked: 40
Hours Worked: 70
Hours Worked: 50
Hours Worked: 50
Hours Worked: 50
```

Cycle 2:

1:

```java
@Test
public void testMain() {
    int shift1, regHours, overHours, rate, overRate, over;

    regHours = 40;
    overHours = 70;
    rate = 50;

    int expResult1A = 2000;
    shift1 = (regHours * rate);

    assertEquals(expResult1A, shift1);
    System.out.println("Shift 1\n - Hours: " + regHours + "\n - Regular Pay: R" + shift1 + "\n - Net Pay: R" + shift1);

    over = overHours - regHours;
    overRate = over * (int) (rate * 1.5);

    int expResult1B = 4250;
    shift1 = (expResult1A + overRate);

    assertEquals(expResult1B, shift1);
    System.out.println("Shift 1\n - Hours: " + overHours + "\n - Overtime Pay: R" + overRate + "\n - Net Pay: R" + shift1);
```

```java
int shift2;
regHours = 40;
overHours = 70;
rate = 70;

int expResult2A = 2800;

shift2 = (regHours * rate);
assertEquals(expResult2A, shift2);
System.out.println("Shift 2\n - Hours: " + regHours + "\n - Regular Pay: R" + shift2 + "\n - Net Pay without Ret: R" + shift2);

over = overHours - regHours;
overRate = over * (int) (rate * 1.5);
int expResult2B = 5950;
shift2 = (expResult2A + overRate);
assertEquals(expResult2B, shift2);
System.out.println("Shift 2\n - Hours: " + overHours + "\n - Overtime Pay: R" + overRate + "\n - Net Pay without Ret: R" + shift2);
```

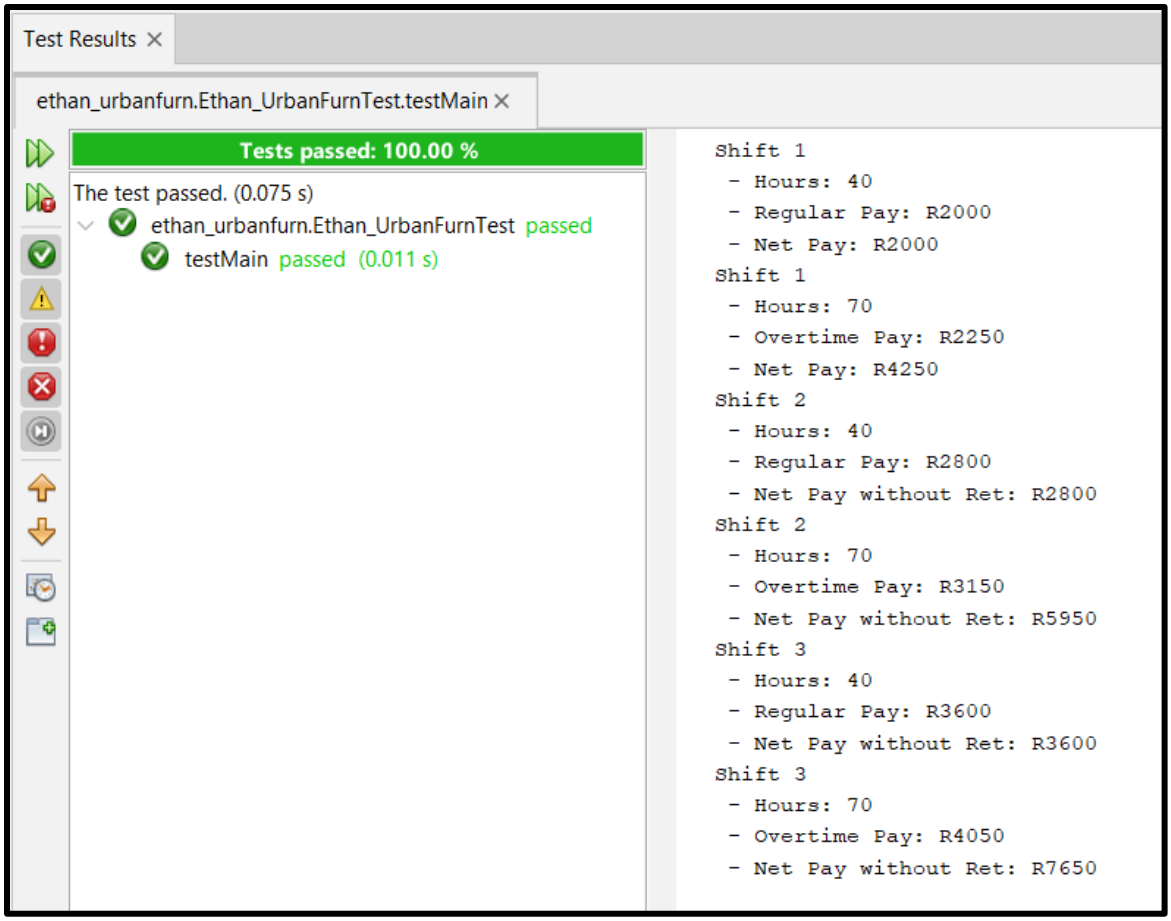
```java
int shift3;
regHours = 40;
overHours = 70;
rate = 90;

int expResult3A = 3600;

shift3 = (regHours * rate);

assertEquals(expResult3A, shift3);
System.out.println("Shift 3\n - Hours: " + regHours + "\n - Regular Pay: R" + shift3 + "\n - Net Pay without Ret: R" + shift3);

over = overHours - regHours;
overRate = over * (int) (rate * 1.5);
int expResult3B = 7650;
shift3 = (expResult3A + overRate);
assertEquals(expResult3B, shift3);
System.out.println("Shift 3\n - Hours: " + overHours + "\n - Overtime Pay: R" + overRate + "\n - Net Pay without Ret: R" + shift3);
```


Test Results
ethan_urbanfurn.Ethan_UrbanFurnTest.testMain
Tests passed: 100.00 %
The test passed. (0.075 s)
ethan_urbanfurn.Ethan_UrbanFurnTest passed
testMain passed (0.011 s)
Shift 1
- Hours: 40
- Regular Pay: R2000
- Net Pay: R2000
Shift 1
- Hours: 70
- Overtime Pay: R2250
- Net Pay: R4250
Shift 2
- Hours: 40
- Regular Pay: R2800
- Net Pay without Ret: R2800
Shift 2
- Hours: 70
- Overtime Pay: R3150
- Net Pay without Ret: R5950
Shift 3
- Hours: 40
- Regular Pay: R3600
- Net Pay without Ret: R3600
Shift 3
- Hours: 70
- Overtime Pay: R4050
- Net Pay without Ret: R7650

2:

```java
@Test
public void testRetire2() {
    int shift2,regHours,rate;
    regHours = 40;
    rate = 70;
    double retPlan;

    int expResult2A = 2800;

    shift2 = (regHours * rate);
    retPlan = (shift2 *0.05);
    double retShift2 = shift2 - retPlan;
    assertEquals(expResult2A, shift2);

    System.out.println("""
                        Regular Shift 2:
                        - Without Retirement: R0.00
                        - With Retirement: R""" + retPlan + "\n- Net Pay with Ret: R" + retShift2);
}
```

Test Results

ethan_urbanfurn.Ethan_UrbanFurnTest.testRetire2

Tests passed: 100.00 %

The test passed. (0.085 s)

ethan_urbanfurn.Ethan_UrbanFurnTest passed

testRetire2 passed (0.012 s)

```
Regular Shift 2:
- Without Retirement: R0.00
- With Retirement: R140.0
- Net Pay with Ret: R2660.0
```

3:

```java
@Test
public void testRetireOver2() {
    int shift2,regHours,rate,overHours,overRate,over;
    regHours = 40;
    rate = 70;
    overHours = 70;
    double retPlan;
    int expResult2A = 2800;

    over = overHours - regHours;
    overRate = over * (int) (rate * 1.5);
    int expResult2B = 5950;
    shift2 = (expResult2A + overRate);
    retPlan = (shift2 *0.05);
    double retShift2 = shift2 - retPlan;
    assertEquals(expResult2B, shift2);

    System.out.println("""
                    Overtime Shift 2:
                    - Without Retirement: R0.00
                    - With Retirement: R""" + retPlan + "\n- Net Pay with Ret: R" + retShift2);
}
```

Test Results ×

ethan_urbanfurn.Ethan_UrbanFurnTest.testRetireOver2 ×

Tests passed: 100.00 %

The test passed. (0.083 s)

- ethan_urbanfurn.Ethan_UrbanFurnTest passed
  - testRetireOver2 passed (0.011 s)

```
Overtime Shift 2:
- Without Retirement: R0.00
- With Retirement: R297.5
- Net Pay with Ret: R5652.5
```

4:

```java
@Test
public void testRetire3() {
    int shift3,regHours,rate;
    regHours = 40;
    rate = 90;
    double retPlan;

    int expResult3A = 3600;

    shift3 = (regHours * rate);
    retPlan = (shift3 *0.05);
    double retShift3 = shift3 - retPlan;
    assertEquals(expResult3A, shift3);

    System.out.println("""
                        Regular Shift 3:
                        - Without Retirement: R0.00
                        - With Retirement: R""" + retPlan + "\n- Net Pay with Ret: R" + retShift3);
}
```

Test Results

ethan_urbanfurn.Ethan_UrbanFurnTest.testRetire3

Tests passed: 100.00 %

The test passed. (0.07 s)

ethan_urbanfurn.Ethan_UrbanFurnTest passed

testRetire3 passed (0.006 s)

```
Regular Shift 3:
- Without Retirement: R0.00
- With Retirement: R180.0
- Net Pay with Ret: R3420.0
```

5:

```java
@Test
public void testRetireOver3() {
    int shift3,regHours,rate,overHours,overRate,over;
    regHours = 40;
    rate = 90;
    overHours = 70;
    double retPlan;

    int expResult3A = 3600;

    over = overHours - regHours;
    overRate = over * (int) (rate * 1.5);
    int expResult3B = 7650;
    shift3 = (expResult3A + overRate);
    retPlan = (shift3 *0.05);
    double retShift3 = shift3 - retPlan;
    assertEquals(expResult3B, shift3);

    System.out.println("""
                        Overtime Shift 3:
                        - Without Retirement: R0.00
                        - With Retirement: R""" + retPlan + "\n- Net Pay with Ret: R" + retShift3);
}
```

Test Results ×

ethan_urbanfurn.Ethan_UrbanFurnTest.testRetireOver3 ×

Tests passed: 100.00 %

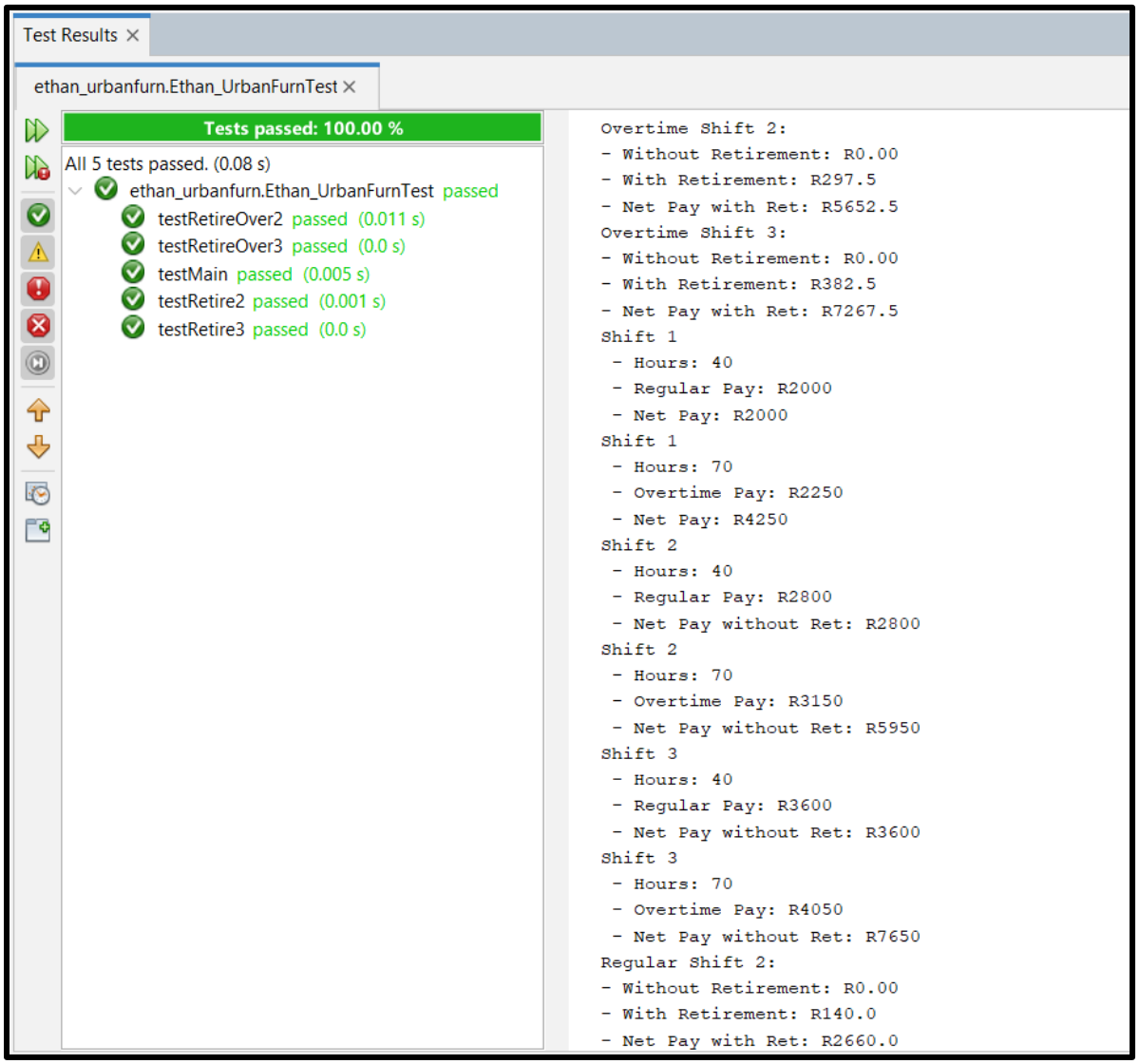
The test passed. (0.075 s)

ethan_urbanfurn.Ethan_UrbanFurnTest passed

testRetireOver3 passed (0.009 s)

```
Overtime Shift 3:
- Without Retirement: R0.00
- With Retirement: R382.5
- Net Pay with Ret: R7267.5
```

6:


Test Results
ethan_urbanfurn.Ethan_UrbanFurnTest
Tests passed: 100.00 %
All 5 tests passed. (0.08 s)
ethan_urbanfurn.Ethan_UrbanFurnTest passed
testRetireOver2 passed (0.011 s)
testRetireOver3 passed (0.0 s)
testMain passed (0.005 s)
testRetire2 passed (0.001 s)
testRetire3 passed (0.0 s)
Overtime Shift 2:
- Without Retirement: R0.00
- With Retirement: R297.5
- Net Pay with Ret: R5652.5
Overtime Shift 3:
- Without Retirement: R0.00
- With Retirement: R382.5
- Net Pay with Ret: R7267.5
Shift 1
- Hours: 40
- Regular Pay: R2000
- Net Pay: R2000
Shift 1
- Hours: 70
- Overtime Pay: R2250
- Net Pay: R4250
Shift 2
- Hours: 40
- Regular Pay: R2800
- Net Pay without Ret: R2800
Shift 2
- Hours: 70
- Overtime Pay: R3150
- Net Pay without Ret: R5950
Shift 3
- Hours: 40
- Regular Pay: R3600
- Net Pay without Ret: R3600
Shift 3
- Hours: 70
- Overtime Pay: R4050
- Net Pay without Ret: R7650
Regular Shift 2:
- Without Retirement: R0.00
- With Retirement: R140.0
- Net Pay with Ret: R2660.0