



FACULTY OF INFORMATION TECHNOLOGY

PROGRAMMING 621 – C++

1ST SEMESTER ASSIGNMENT

Name & Surname: JESHUA LEONARD PILLAY ICAS No: 401914544

Qualification: BSc I.T (2nd Year) Semester: 1 Module Name: PROGRAMMING 621

Date Submitted: 22 / 04 / 2021

ASSESSMENT CRITERIA	MARK ALLOCATION	EXAMINER MARKS	MODERATOR MARKS
MARKS FOR CONTENT			
QUESTION ONE	35		
QUESTION TWO	30		
QUESTION THREE	35		
TOTAL MARKS	100		
Examiner's Comments:			
Moderator's Comments:			
Signature of Examiner:		Signature of Moderator:	

TABLE OF CONTENTS

PAGE NUMBER	CONTENT
2, 3, 4, 5.	QUESTION 1
6, 7, 8, 9.	QUESTION 2
10, 11, 12, 13, 14, 15, 16, 17.	QUESTION 3
18.	REFERENCES

QUESTION 1

In this question, we are required to build a program that performs *division calculations*. We will explain each part of the code and then we will show the entire code with the output at the end of this question.

Before we start programming our division calculator, the first requirement of this program is to make sure all inputted values are positive. We do this by using a Boolean function called “*reduce*” which takes two *float* arguments called *num* and *denom*. This function will either return *true* if the user has inputted all *positive numbers* or the program will return *false* indicating that the inputted numbers were incorrect therefore throwing an error. Below is the Boolean function code which is located outside the *main function*:

```
//Jeshua Leonard Pillay, 401914544

bool reduce(float num, float denom) {

    if(num <= 0 || denom <= 0) {
        return false;
        cout << endl;
    } else if(num >= 1 || denom >= 1) {
        return true;
        cout << endl;
    }

    return num, denom;
}
```

The next part of our code will be a *void function* called *GCD* (greatest common divisor). This function will take the inputted values from the *reduce function* and find the greatest common divisor. This means that, not only will the inputted values be divided but it will display the answer in its simplest fraction form.

```
void GCD(int& num, int& denom) {

    int divisor = 0;

    for(int i = fmin(num, denom) ; i > 0; i--) {
        if(num% i == 0 && denom% i == 0) {
            num /= i;
            denom /= i;
        }
    }
}
```

The next part of our code takes place within the *main function* of our program. We declare *two float variables* called *n* and *d* which stores the input for the numerator and denominator respectively. Below is the code for getting the input from the user:

```
//Jeshua Leonard Pillay, 401914544
```

```
float n, d;  
cout << "Enter numerator: ";  
cin >> n;  
cout << "Enter denominator: ";  
cin >> d;
```

Lastly, in our *main function* we have an *if-statement*. This conditional statement will take the inputted values and test it with the previously explained *Boolean function* called *reduce* and *GCD*. If the Boolean function returns *false* then it will alert the user that there has been an error and if it returns *true* then the program will proceed with the calculation. Below is the code for the conditional statement and the calculation:

```
//Jeshua Leonard Pillay, 401914544
```

```
if(reduce(n, d) == false) {  
    cout << "Fraction Error! All inputted values must be above zero.\n" << endl;  
} else if(reduce(n, d) == true) {  
    GCD(n, d);  
    cout << "Your answer is: " << n << "/" << d << endl;  
}
```

We have explained the entire code and below is what the entire code looks like:

```
#include <iostream>  
#include <cmath>  
//Jeshua Leonard Pillay, 401914544  
  
using namespace std;  
  
bool reduce(float num, float denom) {  
  
    if(num <= 0 || denom <= 0) {  
        return false;  
        cout << endl;  
    } else if(num >= 1 || denom >= 1) {  
        return true;  
        cout << endl;  
    }  
  
    return num, denom;  
}  
  
void GCD(int& num, int& denom) {  
  
    int divisor = 0;  
  
    for(int i = fmin(num, denom) ; i > 0; i--) {  
        if(num% i == 0 && denom% i == 0) {
```

```

        num /= i;
        denom /= i;
    }
}

int main() {

    cout << "Jeshua Leonard Pillay, 401914544" << endl;

    int n, d;
    cout << endl << "Enter numerator: ";
    cin >> n;
    cout << "Enter denominator: ";
    cin >> d;

    if(reduce(n, d) == false) {
        cout << "Fraction Error! All inputted values must be above zero.\a" << endl;
    } else if(reduce(n, d) == true) {
        GCD(n, d);
        cout << "Your answer is: " << n << "/" << d << endl;
    }

    return 0;
}

```

We will now run the program and input **25** as the *numerator* (n) and **15** as the *denominator* (d):

The screenshot shows the Visual Studio Code editor with a C++ file named `Q1.cpp`. The code is as follows:

```

1 #include <iostream>
2 #include <cmath>
3 //Jeshua Leonard Pillay, 401914544
4
5 using namespace std;
6
7 bool reduce(float num, float denom) {
8
9     if(num <= 0 || denom <= 0) {
10         return false;
11         cout << endl;
12     } else if(num >= 1 || denom >= 1) {
13         return true;
14         cout << endl;
15     }
16
17     return num, denom;
18 }
19
20 void GCD(int& num, int& denom) {...
21 }
22
23 int main() {
24     cout << "Jeshua Leonard Pillay, 401914544" << endl;
25
26     int n, d;
27     cout << endl << "Enter numerator: ";
28     cin >> n;
29     cout << "Enter denominator: ";
30     cin >> d;
31
32     if(reduce(n, d) == false) {
33         cout << "Fraction Error! All inputted values must be above zero.\a" << endl;
34     } else if(reduce(n, d) == true) {
35         GCD(n, d);
36         cout << "Your answer is: " << n << "/" << d << endl;
37     }
38 }

```

The terminal output shows the program's execution:

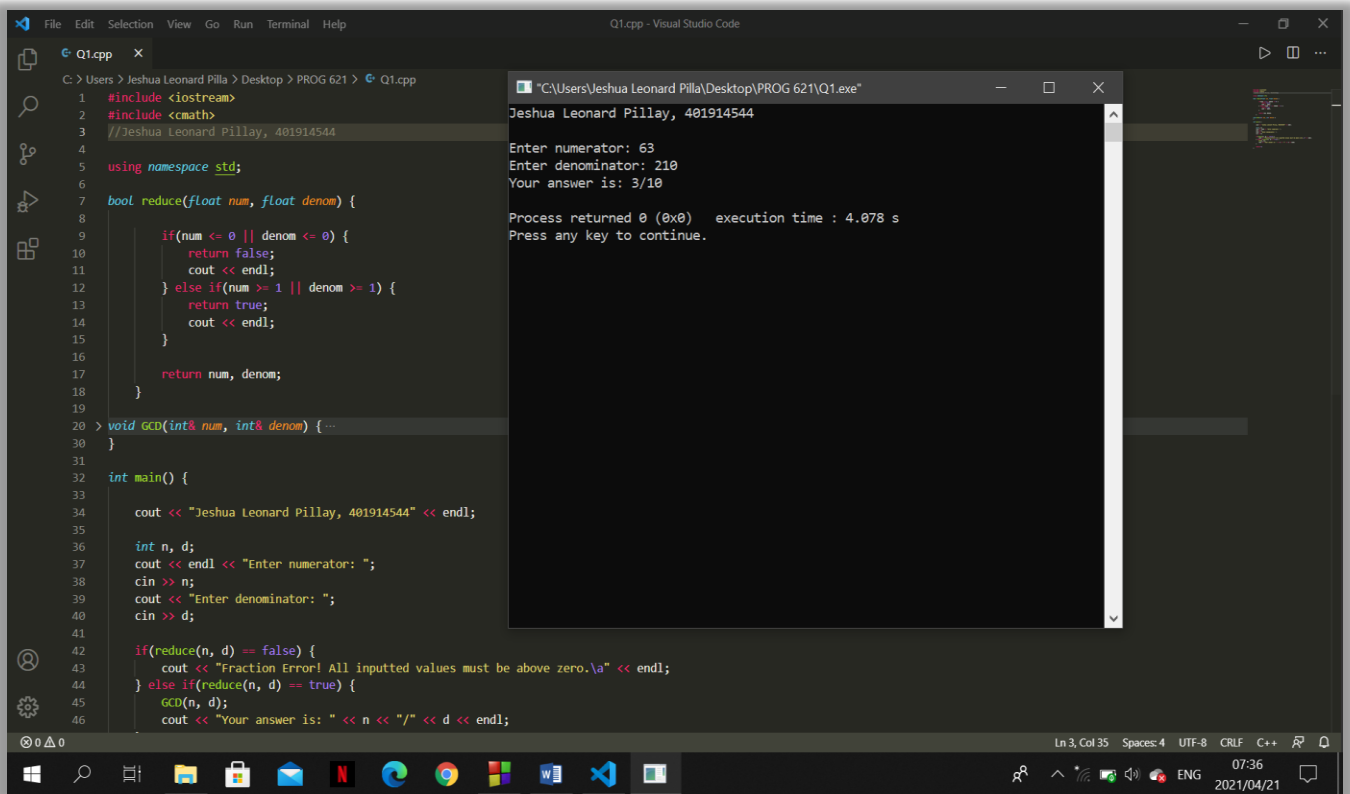
```

Jeshua Leonard Pillay, 401914544
Enter numerator: 25
Enter denominator: 15
Your answer is: 5/3

Process returned 0 (0x0)   execution time : 4.547 s
Press any key to continue.

```

We will now run the program again and input **63** as the **numerator** (n) and **210** as the **denominator** (d):



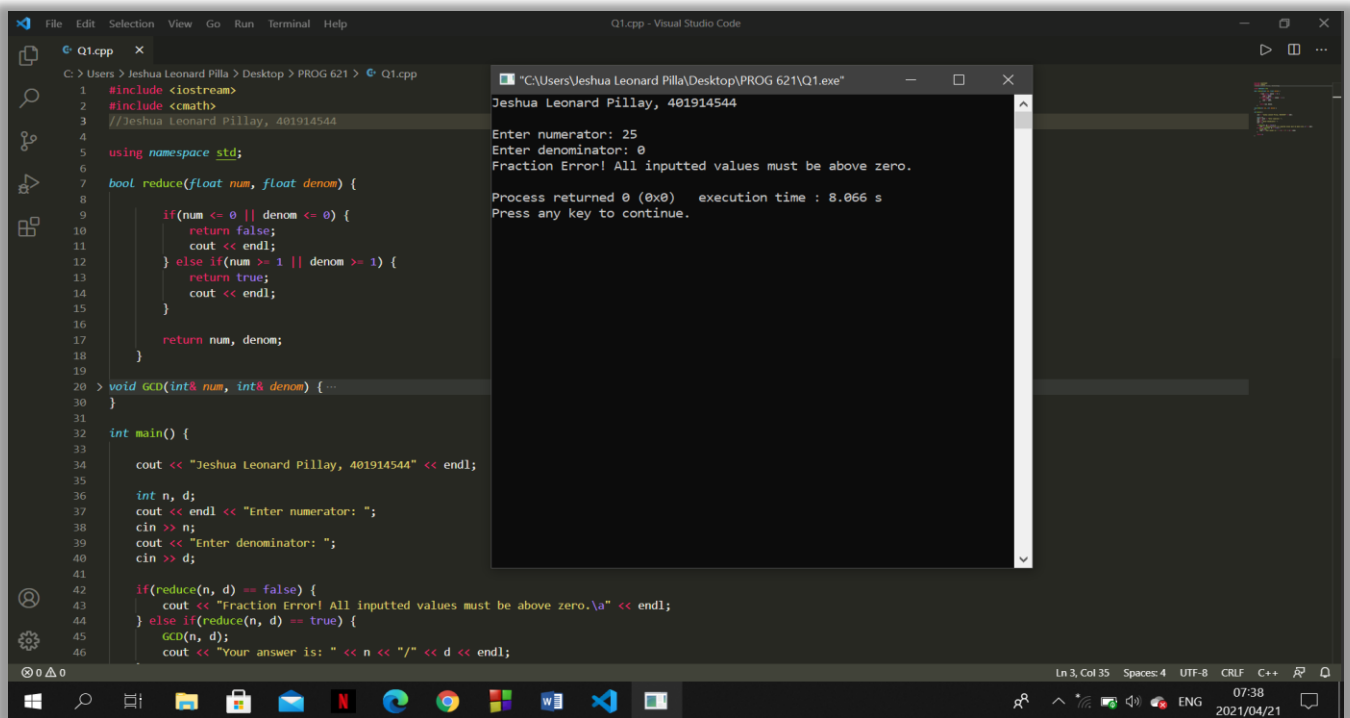
```
Q1.cpp
1 #include <iostream>
2 #include <cmath>
3 //Jeshua Leonard Pillay, 401914544
4
5 using namespace std;
6
7 bool reduce(float num, float denom) {
8
9     if(num <= 0 || denom <= 0) {
10         return false;
11         cout << endl;
12     } else if(num >= 1 || denom >= 1) {
13         return true;
14         cout << endl;
15     }
16
17     return num, denom;
18 }
19
20 void GCD(int& num, int& denom) { ...
21 }
22
23 int main() {
24     cout << "Jeshua Leonard Pillay, 401914544" << endl;
25
26     int n, d;
27     cout << endl << "Enter numerator: ";
28     cin >> n;
29     cout << "Enter denominator: ";
30     cin >> d;
31
32     if(reduce(n, d) == false) {
33         cout << "Fraction Error! All inputted values must be above zero.\a" << endl;
34     } else if(reduce(n, d) == true) {
35         GCD(n, d);
36         cout << "Your answer is: " << n << "/" << d << endl;
37     }
38 }
```

Output:

```
Jeshua Leonard Pillay, 401914544
Enter numerator: 63
Enter denominator: 210
Your answer is: 3/10

Process returned 0 (0x0)   execution time : 4.078 s
Press any key to continue.
```

We will now run the program one last time. In this case, we will input **25** as the **numerator** (n) and **0** as the **denominator** (d). The program will alert the user with an error:



```
Q1.cpp
1 #include <iostream>
2 #include <cmath>
3 //Jeshua Leonard Pillay, 401914544
4
5 using namespace std;
6
7 bool reduce(float num, float denom) {
8
9     if(num <= 0 || denom <= 0) {
10         return false;
11         cout << endl;
12     } else if(num >= 1 || denom >= 1) {
13         return true;
14         cout << endl;
15     }
16
17     return num, denom;
18 }
19
20 void GCD(int& num, int& denom) { ...
21 }
22
23 int main() {
24     cout << "Jeshua Leonard Pillay, 401914544" << endl;
25
26     int n, d;
27     cout << endl << "Enter numerator: ";
28     cin >> n;
29     cout << "Enter denominator: ";
30     cin >> d;
31
32     if(reduce(n, d) == false) {
33         cout << "Fraction Error! All inputted values must be above zero.\a" << endl;
34     } else if(reduce(n, d) == true) {
35         GCD(n, d);
36         cout << "Your answer is: " << n << "/" << d << endl;
37     }
38 }
```

Output:

```
Jeshua Leonard Pillay, 401914544
Enter numerator: 25
Enter denominator: 0
Fraction Error! All inputted values must be above zero.

Process returned 0 (0x0)   execution time : 8.066 s
Press any key to continue.
```

This program was a good introduction into our assignment as it tested important foundational concepts such as obtaining input from a user, using functions and conditional statements to process the input and then simply displaying the output.

QUESTION 2

In this question, we are required to build a **parking ticket system** for a parking garage. The requirements for this program are as follows:

- ❖ The garage will charge a **minimum** fee of **R12** for parking between **1 – 3 hours**.
- ❖ An **additional 0.90c** will be charged for every hour parked **after** 3 hours.
- ❖ The **maximum** charge for a 24 hour period is **R24**.
- ❖ The program will contain information based on **three** customers only.
- ❖ The program must display output in a neat **tabular format**.

Just like the previous question, we will explain each part of the code to see how it all works together and at the end of the question we will show the entire code along with the outputs. We will begin coding with a function of type **float** called **calculateCharges** located outside the **main function**. This function will contain an argument called **hrs** which will be of type float as well. In this code, we have another variable of type **float** called **cost** which will have a value of **12**. Below is the code for the function:

```
float calculateCharge(float hrs) {  
  
    float cost = 12.0;  
  
    if(hrs > 3 && hrs < 12) {  
        cost += 0.9*(hrs - 3);  
    } else if(hrs >= 12) {  
        cost = 20.0;  
    }  
  
    return cost;  
}
```

We now move to our **int main function**. We begin by creating a **2D array** called **table** containing **three rows** and **two columns** and they will all have an **initial** value of **0,0**. We will then create **three float** variables called **hours**, **totalHours** and **totalCharge** of which totalHours and totalCharge will each have an **initial** value of **0,0**.

```
float table[3][2] = {  
    {0.0, 0.0},  
    {0.0, 0.0},  
    {0.0, 0.0}  
};  
  
float hours;  
  
float totalHours = 0.0;  
float totalCharge = 0.0;
```

The next part of our code is the most crucial. We will have a **for-loop** prompting the user to enter the number of **hours** three times and it will be stored in the **first** column of the 2D array. The variable totalHours will

then add up all the inputted hours per row. All hours inputted in the first column will then be stored in the second column with a previously explained function called *calculateCharge* which will then use the hours inputted to calculate the charge per customer. Lastly, we have the variable called *totalCharge* which will add up all the charges calculated per row. The code is found below:

```
// Jeshua Leonard Pillay, 401914544
for(int i = 0; i < 3; i++) {
    cout << "Enter hours: ";
    cin >> hours;

    table[i][0] = hours;
    totalHours += hours;

    table[i][1] = calculateCharge(hours);
    totalCharge += calculateCharge(hours);
}
```

This line of code is simply used to display our headers for each columns:

```
// Jeshua Leonard Pillay, 401914544
cout << "Car:\tHours:\tCharge:" << endl;
```

Next we have a *for-loop* which is simply used to number each row:

```
// Jeshua Leonard Pillay, 401914544
for(int i = 0; i < 3; i++) {
    cout << i + 1 << "\t" << table[i][0] << "\t" << table[i][1] << endl;
}
```

Our last line of code will output all the hours inputted and charges calculated.

```
// Jeshua Leonard Pillay, 401914544
cout << "Total: \t" << totalHours << "\t" << totalCharge;
```


We will now look at the entire code:

```
#include <iostream>
#include <cmath>
#include <string>

using namespace std;

// Jeshua Leonard Pillay, 401914544

float calculateCharge(float hrs) {

    float cost = 12.0;

    if(hrs > 3 && hrs < 12) {
        cost += 0.9*(hrs - 3);
    } else if(hrs >= 12) {
        cost = 20.0;
    }

    return cost;
}

int main() {

    cout << "Jeshua Leonard Pillay, 401914544" << endl;

    float table[3][2] = {
        {0.0, 0.0},
        {0.0, 0.0},
        {0.0, 0.0}
    };

    float hours;

    float totalHours = 0.0;
    float totalCharge = 0.0;

    for(int i = 0; i < 3; i++) {
        cout << endl << "Enter hours: ";
        cin >> hours;

        table[i][0] = hours;
        totalHours += hours;

        table[i][1] = calculateCharge(hours);
        totalCharge += calculateCharge(hours);
    }

    cout << endl << "Car:\tHours:\tCharge:" << endl;

    for(int i = 0; i < 3; i++) {
```

```

        cout << i + 1 << "\t" << table[i][0] << "\t" << table[i][1] << endl;
    }

    cout << "Total: \t" << totalHours << "\t" << totalCharge;

    return 0;
}

```

We will now run the program and input hours of **1.5**, **4** and **24**. Any input that is greater than 12 hours will have a charge of R20 as that is the **maximum** charge for a 24 hour period.

```

// Q2.cpp
1 #include <iostream>
2 #include <cmath>
3 #include <string>
4
5 using namespace std;
6
7 float calculateCharge(float hrs) {
8
9     float cost = 12.0;
10
11     if(hrs > 3 && hrs < 12) {
12         cost += 0.9*(hrs - 3);
13     } else if(hrs >= 12) {
14         cost = 20.0;
15     }
16
17     return cost;
18 }
19
20 int main() {
21
22     // Jeshua Leonard Pillay, 401914544
23
24     cout << "Jeshua Leonard Pillay, 401914544" << endl;
25
26     float table[3][2] = {
27         {0.0, 0.0},
28         {0.0, 0.0},
29         {0.0, 0.0}
30     };
31
32     float hours;
33
34     float totalHours = 0.0;
35     float totalCharge = 0.0;
36
37     for(int i = 0; i < 3; i++) {

```

```

C:\Users\Jeshua Leonard Pillay\Desktop\PROG 621\Q2.exe
Jeshua Leonard Pillay, 401914544
Enter hours: 1.5
Enter hours: 4
Enter hours: 24

Car:   Hours:  Charge:
1      1.5     12
2      4       12.9
3      24      20
Total: 29.5    44.9
Process returned 0 (0x0)   execution time : 18.584 s
Press any key to continue.

```

This question was a step up from the previous question as it not only tested the same concepts but a new additional concept such as a 2D array to display all input in a tabular format.

QUESTION 3

We have now arrived at our last question. In this question we are required to program a payroll system. Before we begin programming, let us look at the program requirements:

- ❖ We must declare a base class called *Emp*.
- ❖ Use a function called *getInfo* to get the employee details.
- ❖ Declare a derived class called *Salary*.
- ❖ Use a function called *getSalary* to get salary details of employee.
- ❖ Create another function called *calculateNet* to find the employee's net pay.

Now that we have established the program's requirements, we can now start coding. We will begin by declaring the *base class Emp* and it will be set to *public*. Within this class we will create a function called *getInfo* of type *string* and we will pass an argument of type *integer* called *Emp_ID*. Within this function we will create a variable of type *string* called *Emp_Det*.

We will then create a *switch statement* and pass the *Emp_ID* argument. Each *case* will have a unique number which will actually be our employee's ID. *Emp_Det* will store different employee details depending on which employee ID the user inputs for. We will also include a *default* option if the user inputs incorrectly. Below is the code for the *class Emp*.

```
class Emp{
// Jeshua Leoanrd Pillay, 401914544

public:
    string getInfo(int Emp_ID) {
        string Emp_Det;

        switch(Emp_ID) {

            case 1104:
                Emp_Det = "Employee's ID: 1104\nFirstname: Jesh\nSurname: Pillay\nage: 20\n";
                break;

            case 1234:
                Emp_Det = "Employee's ID: 1234\nFirstname: Shai\nSurname: Dookan\nage: 19\n";
                break;

            case 2424:
                Emp_Det = "Employee's ID: 2424\nFirstname:Adin\nSurname: Rikisahedew\nage: 19\n";
                break;

            default:
                Emp_Det = "Invalid input! \a";
        }

        return Emp_Det;
    }
};
```

We will now look at the *salary class*. It is very similar to *class Emp* but with a few additions. We will inherit the employee class and set it to *public* as well. We will then declare *six float variables* called *GP* (gross profit), *hours*, *days*, *rate*, *tax* and *NP* (net profit) which we will look at a later stage.

Now we will create a *getSalary function* of type *string* and pass an argument of type *integer* called *Emp_ID*. Within this function we will create a variable of type *string* called *Salary_Det*. We will then create a *switch statement* and pass the *Emp_ID* argument. Each *case* will have a unique number which will actually be our employee's ID. *Sal_Det* will store different employee salary details depending on which employee ID the user inputs for. We will also include a *default* option if the user inputs incorrectly. Below is the code for the *class Salary*.

```
// Jeshua Leoanrd Pillay, 401914544
class Salary: public Emp{
    public:
        float GP, hours, days, rate, tax, NP;
        string getSalary(int Emp_ID) {
            string Salary_Det;

            switch(Emp_ID) {

                case 1104:
                    Salary_Det = "Employee's Full Name: Jesh Pillay\nHourly rate: R500";
                    break;

                case 1234:
                    Salary_Det = "Employee's Full Name: Shai Dookan\nHourly rate: R600";
                    break;

                case 2424:
                    Salary_Det = "Employee' Full Name: Adin Rikisahedew\nHourly rate: R700";
                    break;

                default:
                    Salary_Det = "Invalid input! \a";

            }

            return Salary_Det;
        }
}
```

Within the salary class, we will have another function of type *integer* called *calculateNet* and we will pass an argument of type *integer* called *Emp_ID*. Within this function we will then create a *switch statement* and pass the *Emp_ID* variable. Just like the previous switch statement *cases*, it will have unique case numbers which will be the employee's ID and it will only display information relating to the specific ID the user inputs.

Outside the switch statement we will now use the six previously declared float variables that we mentioned earlier. We will ask the user to input the *rate per hour*, *hours worked* and *number of days work*. A calculation will be done to get the gross profit and tax will be deducted to get the net profit. Below is the code for the function:

```
// Jeshua Leoanrd Pillay, 401914544
int calculateNet(int Emp_ID) {

    switch(Emp_ID) {
        case 1104:
            cout << endl << "Employee's Full Name: Jesh Pillay\n";
            break;

        case 1234:
            cout << endl << "Employee's Full Name: Shai Dookan\n";
            break;

        case 2424:
            cout << endl << "Employee's Full Name: Adin Rikisahedew\n";
            break;

        default:
            cout << endl << "Invalid input! \a";
            break;
    }

    cout << endl << "Enter the rate per hour: R";
    cin >> rate;
    cout << endl << "Enter the hours: ";
    cin >> hours;
    cout << endl << "Enter the number of days worked: ";
    cin >> days;
    GP = rate * hours * days;
    tax = 0.15;
    NP = GP - (GP * tax);

    return NP;
}

};
```

We now move to the *int main function*. In order to get the employee's details, we will now call the class *Emp* with *emp_det*. We will create a variable of type *integer* called *ID*. We will ask the user to input an ID and it will get stored in the *getInfo function*. Below is the code for getting employee's details:

```
Emp emp_det;

// Jeshua Leoanrd Pillay, 401914544
int ID;
cout << endl << "Enter employee's ID number to view employee's details: ";
cin >> ID;
cout << endl << emp_det.getInfo(ID) << endl;
```

We will do the same for getting employee salary details and calculating net pay:

```
Salary sal_det;
```

```

int ID2;
cout << endl << "Enter employee's ID number to view salary details: ";
cin >> ID2;
cout << endl << sal_det.getSalary(ID2) << endl;
int ID3;
cout << endl << "Enter employee's ID number to calculate net payment: ";
cin >> ID3;
cout << endl << "Employee's net payment (tax included) is : R" << sal_det.calculateNet
(ID3) << endl;

```

Let us now look at the entire code:

```

#include <iostream>
#include <cmath>

using namespace std;

// Jeshua Leoanrd Pillay, 401914544
class Emp{

public:
    string getInfo(int Emp_ID) {
        string Emp_Det;

        switch(Emp_ID) {

            case 1104:
                Emp_Det = "Employee's ID: 1104\nFirstname: Jesh\nSurname: Pillay\nage: 20\n";
                break;

            case 1234:
                Emp_Det = "Employee's ID: 1234\nFirstname: Shai\nSurname: Dookan\nage: 19\n";
                break;

            case 2424:
                Emp_Det = "Employee's ID: 2424\nFirstname:Adin\nSurname: Rikisahedew\nage: 19\n";
                break;

            default:
                Emp_Det = "Invalid input! \a";
        }

        return Emp_Det;
    }

};

// Jeshua Leoanrd Pillay, 401914544
class Salary: public Emp{
public:
    float GP, hours, days, rate, tax, NP;

```

```

    string getSalary(int Emp_ID) {
        string Salary_Det;

        switch(Emp_ID) {

            case 1104:
                Salary_Det = "Employee's Full Name: Jesh Pillay\nHourly rate: R500";
                break;

            case 1234:
                Salary_Det = "Employee's Full Name: Shai Dookan\nHourly rate: R600";
                break;

            case 2424:
                Salary_Det = "Employee' Full Name: Adin Rikisahedew\nHourly rate: R700";
                break;

            default:
                Salary_Det = "Invalid input! \a";

        }

        return Salary_Det;
    }

    // Jeshua Leoanrd Pillay, 401914544
    int calculateNet(int Emp_ID) {

        switch(Emp_ID) {
            case 1104:
                cout << endl << "Employee's Full Name: Jesh Pillay\n";
                break;

            case 1234:
                cout << endl << "Employee's Full Name: Shai Dookan\n";
                break;

            case 2424:
                cout << endl << "Employee's Full Name: Adin Rikisahedew\n";
                break;

        }

        cout << endl << "Enter rate per hour: R";
        cin >> rate;
        cout << endl << "Enter the hours worked: ";
        cin >> hours;
        cout << endl << "Enter the number of days worked: ";
        cin >> days;
        GP = rate * hours * days;
        tax = 0.15;
        NP = GP - (GP * tax);

        return NP;
    }

```

```

};

int main() {
// Jeshua Leoanrd Pillay, 401914544
    cout << "Jeshua Leonard Pillay, 401914544" << endl;

    cout << "....." << endl;

    Emp emp_det;

// Jeshua Leoanrd Pillay, 401914544
    int ID;
    cout << endl << "Enter employee's ID number to view employee's details: ";
    cin >> ID;
    cout << endl << emp_det.getInfo(ID) << endl;

    cout << "....." << endl;

// Jeshua Leoanrd Pillay, 401914544
    Salary sal_det;

    int ID2;
    cout << endl << "Enter employee's ID number to view salary details: ";
    cin >> ID2;
    cout << endl << sal_det.getSalary(ID2) << endl;

    cout << "....." << endl;

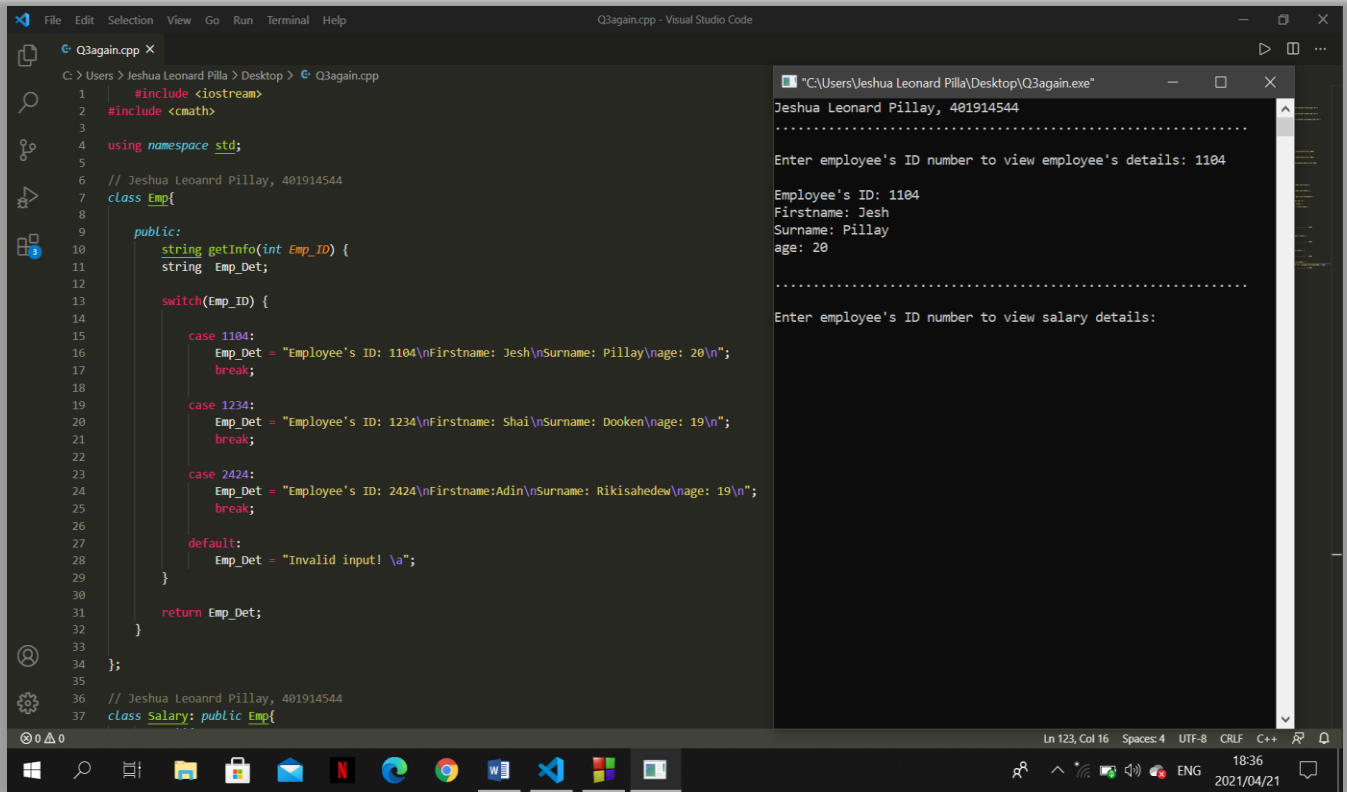
// Jeshua Leoanrd Pillay, 401914544
    int ID3;
    cout << endl << "Enter employee's ID number to calculate net payment: ";
    cin >> ID3;
    cout << endl << "Employee's net payment (tax included) is : R" << sal_det.calculateNet
(ID3) << endl;

    cout << "....." << endl;

    return 0;
}

```


Let us now run the program to show the output. The program will first ask the user to employee's ID and when entered it will display that specific employee. Thereafter, the program will again ask for the employee's ID to view salary details.

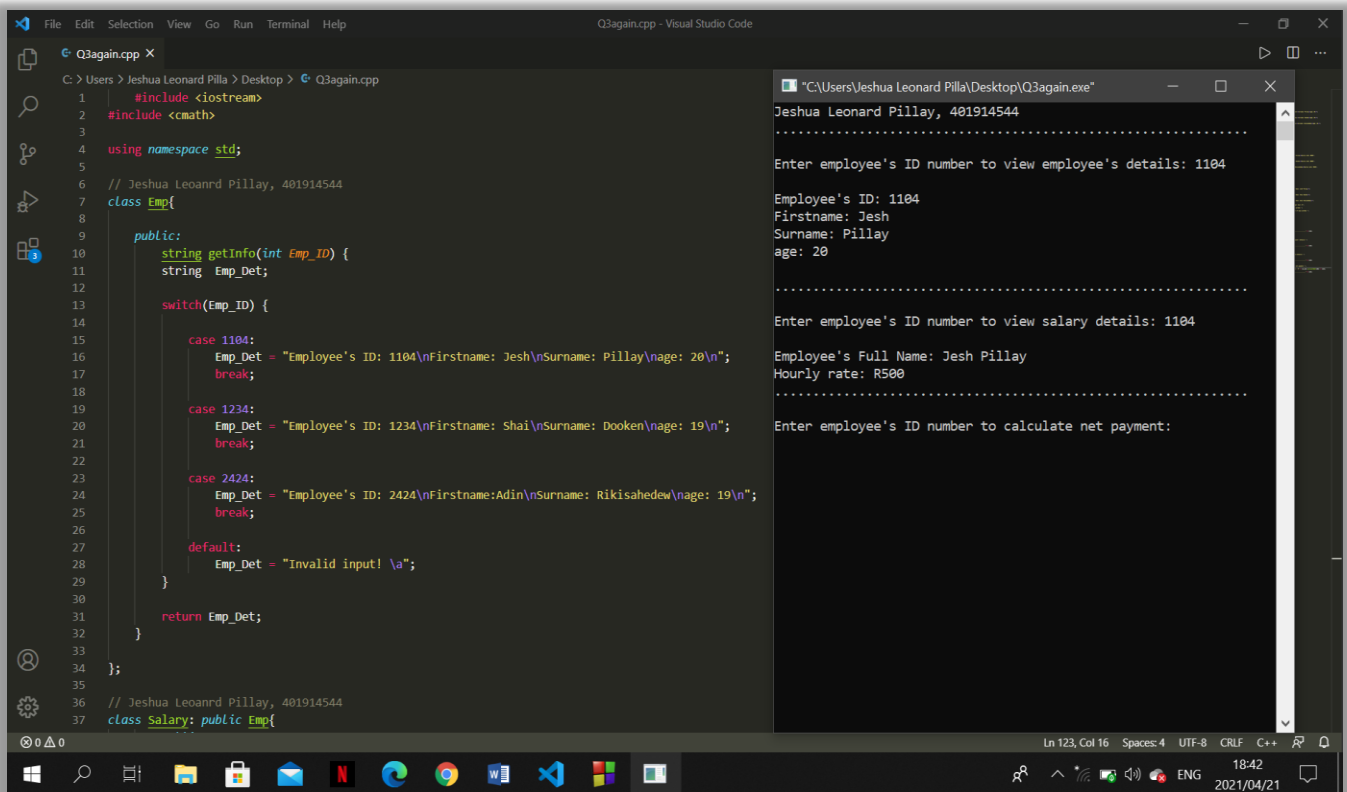


```
Q3again.cpp X
C:\Users\Jeshua Leonard Pillay\Desktop> C:\Q3again.cpp
1 #include <iostream>
2 #include <cmath>
3
4 using namespace std;
5
6 // Jeshua Leonanrd Pillay, 401914544
7 class Emp{
8
9 public:
10 string getInfo(int Emp_ID) {
11     string Emp_Det;
12
13     switch(Emp_ID) {
14
15     case 1104:
16         Emp_Det = "Employee's ID: 1104\nFirstname: Jesh\nSurname: Pillay\nage: 20\n";
17         break;
18
19     case 1234:
20         Emp_Det = "Employee's ID: 1234\nFirstname: Shai\nSurname: Dookan\nage: 19\n";
21         break;
22
23     case 2424:
24         Emp_Det = "Employee's ID: 2424\nFirstname:Adin\nSurname: Rikisahedew\nage: 19\n";
25         break;
26
27     default:
28         Emp_Det = "Invalid input! \a";
29
30     }
31
32     return Emp_Det;
33 };
34
35 // Jeshua Leonanrd Pillay, 401914544
36 class Salary: public Emp{
37
```

```
"C:\Users\Jeshua Leonard Pillay\Desktop\Q3again.exe"
Jeshua Leonard Pillay, 401914544
.....
Enter employee's ID number to view employee's details: 1104

Employee's ID: 1104
Firstname: Jesh
Surname: Pillay
age: 20
.....
Enter employee's ID number to view salary details:
```

When the user enters the employee's ID, it will display the employee's salary details. Thereafter it will ask the user to input Employee's ID to calculate net pay.



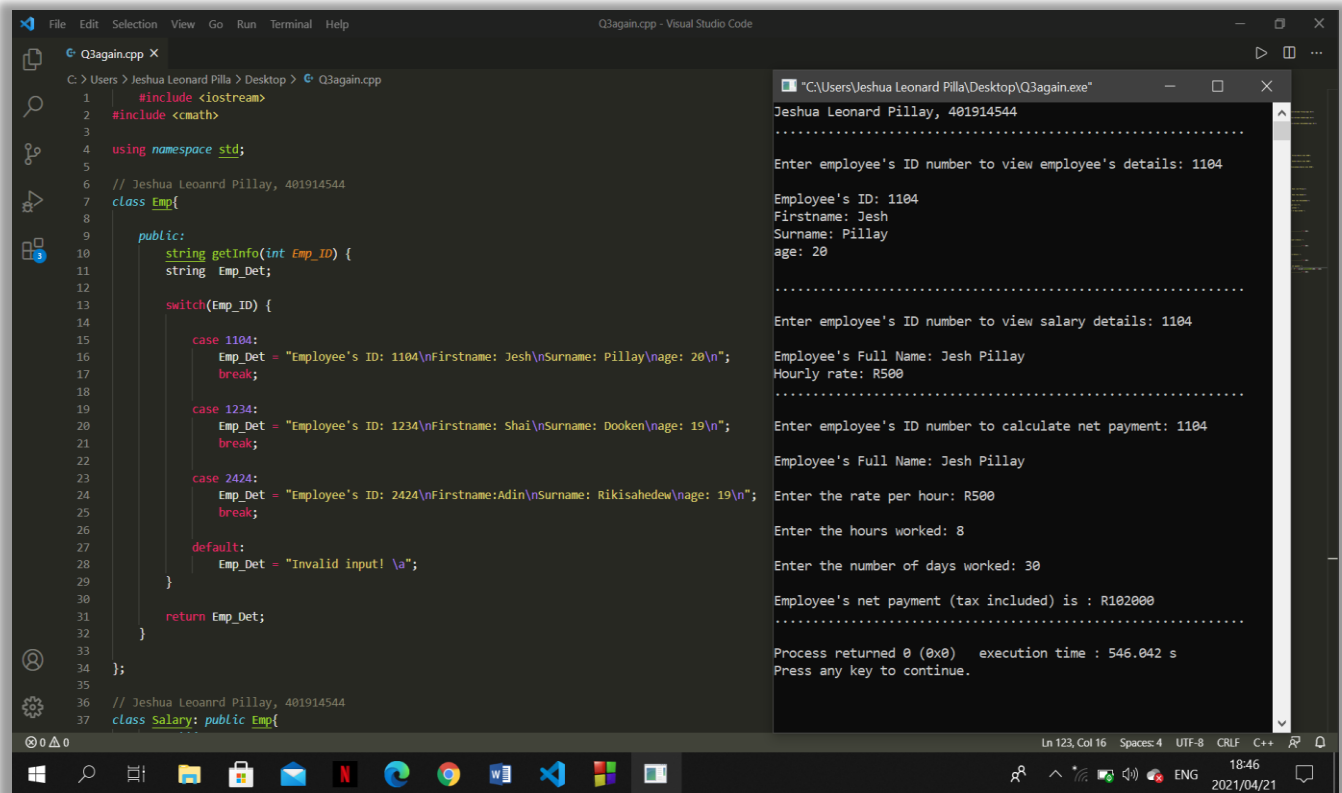
```
Q3again.cpp X
C:\Users\Jeshua Leonard Pillay\Desktop> C:\Q3again.cpp
1 #include <iostream>
2 #include <cmath>
3
4 using namespace std;
5
6 // Jeshua Leonanrd Pillay, 401914544
7 class Emp{
8
9 public:
10 string getInfo(int Emp_ID) {
11     string Emp_Det;
12
13     switch(Emp_ID) {
14
15     case 1104:
16         Emp_Det = "Employee's ID: 1104\nFirstname: Jesh\nSurname: Pillay\nage: 20\n";
17         break;
18
19     case 1234:
20         Emp_Det = "Employee's ID: 1234\nFirstname: Shai\nSurname: Dookan\nage: 19\n";
21         break;
22
23     case 2424:
24         Emp_Det = "Employee's ID: 2424\nFirstname:Adin\nSurname: Rikisahedew\nage: 19\n";
25         break;
26
27     default:
28         Emp_Det = "Invalid input! \a";
29
30     }
31
32     return Emp_Det;
33 };
34
35 // Jeshua Leonanrd Pillay, 401914544
36 class Salary: public Emp{
37
```

```
"C:\Users\Jeshua Leonard Pillay\Desktop\Q3again.exe"
Jeshua Leonard Pillay, 401914544
.....
Enter employee's ID number to view employee's details: 1104

Employee's ID: 1104
Firstname: Jesh
Surname: Pillay
age: 20
.....
Enter employee's ID number to view salary details: 1104

Employee's Full Name: Jesh Pillay
Hourly rate: R500
.....
Enter employee's ID number to calculate net payment:
```

After inputting the employee's ID, it will display the employee's name and then ask the user to input the rate per hour, hours worked and the number of days worked. The program will take the input and calculate the net pay.



```
Q3again.cpp - Visual Studio Code
C:\Users\Jeshua Leonard Pillay\Desktop> C:\Q3again.cpp
1 #include <iostream>
2 #include <cmath>
3
4 using namespace std;
5
6 // Jeshua Leonard Pillay, 401914544
7 class Emp{
8
9 public:
10 string getInfo(int Emp_ID) {
11 string Emp_Det;
12
13 switch(Emp_ID) {
14
15 case 1104:
16 Emp_Det = "Employee's ID: 1104\nFirstname: Jesh\nSurname: Pillay\nage: 20\n";
17 break;
18
19 case 1234:
20 Emp_Det = "Employee's ID: 1234\nFirstname: Shai\nSurname: Dookan\nage: 19\n";
21 break;
22
23 case 2424:
24 Emp_Det = "Employee's ID: 2424\nFirstname: Adin\nSurname: Rikisahedew\nage: 19\n";
25 break;
26
27 default:
28 Emp_Det = "Invalid input! \n";
29
30 }
31 return Emp_Det;
32 }
33 };
34
35 // Jeshua Leonard Pillay, 401914544
36 class Salary: public Emp{
37
```

```
"C:\Users\Jeshua Leonard Pillay\Desktop\Q3again.exe"
Jeshua Leonard Pillay, 401914544
.....
Enter employee's ID number to view employee's details: 1104
.....
Employee's ID: 1104
Firstname: Jesh
Surname: Pillay
age: 20
.....
Enter employee's ID number to view salary details: 1104
.....
Employee's Full Name: Jesh Pillay
Hourly rate: R500
.....
Enter employee's ID number to calculate net payment: 1104
.....
Employee's Full Name: Jesh Pillay
Enter the rate per hour: R500
Enter the hours worked: 8
Enter the number of days worked: 30
.....
Employee's net payment (tax included) is : R102000
.....
Process returned 0 (0x0)   execution time : 546.042 s
Press any key to continue.
```

We have finally completed the last question of our assignment. This question tested a variety of concepts such as classes, functions, inheritance, conditional statements, getting the input, displaying the output and mathematical calculations.

REFERENCES

Augustine, R. (2021, February 23). *C++ If ... Else*. Retrieved from w3schools:
https://www.w3schools.com/cpp/cpp_conditions.asp

Cooper, J. (2020, August 21). *C++ Function Parameters*. Retrieved from w3schools:
https://www.w3schools.com/cpp/cpp_function_param.asp

Jenkins, W. (2020, December 21). *C++ For Loop*. Retrieved from w3schools:
https://www.w3schools.com/cpp/cpp_for_loop.asp

Norman, O. (2021, January 13). *C++ Switch*. Retrieved from w3schools:
https://www.w3schools.com/cpp/cpp_switch.asp

Rose, J. (2020, November 12). *C++ Classes and Objects*. Retrieved from w3schools:
https://www.w3schools.com/cpp/cpp_classes.asp

Rowan, A. (2021, March 25). *C++ Variables*. Retrieved from w3schools:
https://www.w3schools.com/cpp/cpp_variables.asp

Veltman, R. V. (2020, December 15). *C++ Arrays*. Retrieved from w3schools:
https://www.w3schools.com/cpp/cpp_arrays.asp