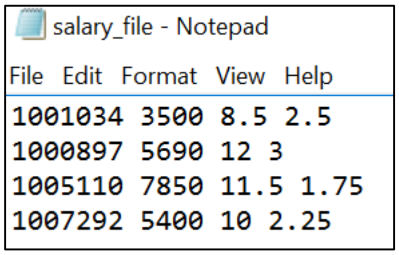
LAB OBJECTIVE

At the end of this lab activity, the students should be able to:

- Read and write files to solve programming problems.

PRACTICE

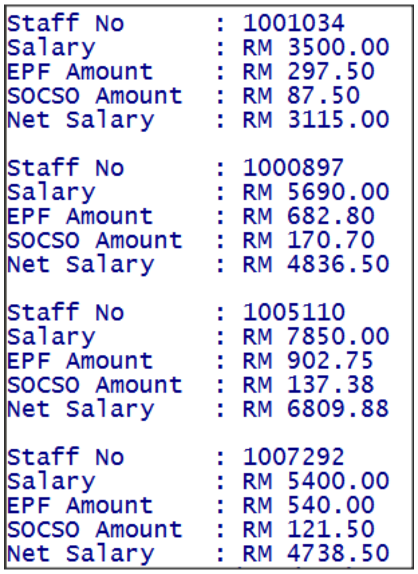
1. Create a text file called *salary_file.txt*. The file has the following data which is the *staff ID*, *salary*, *epf percentage* and *socso percentage*.



```
1001034 3500 8.5 2.5
1000897 5690 12 3
1005110 7850 11.5 1.75
1007292 5400 10 2.25
```

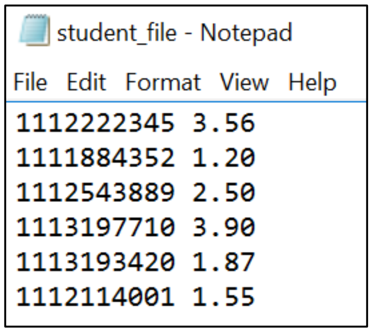
In the main() function:

- Create a FILE pointer called *fread*. Use this pointer to open the file *salary_file.txt* for reading.
- If the file cannot be opened, display error message *"File cannot be accessed!"* and quit the program.
- Read all the data from the file.
- Calculate the epf deduction amount (epf percentage/100 x salary).
- Calculate the socso deduction amount (socso percentage/100 x salary).
- Calculate the new salary after all deductions.
- Display the output on the screen as shown below.



```
Staff No       : 1001034
Salary         : RM 3500.00
EPF Amount     : RM 297.50
SOCSO Amount   : RM 87.50
Net Salary     : RM 3115.00

Staff No       : 1000897
Salary         : RM 5690.00
EPF Amount     : RM 682.80
SOCSO Amount   : RM 170.70
Net Salary     : RM 4836.50

Staff No       : 1005110
Salary         : RM 7850.00
EPF Amount     : RM 902.75
SOCSO Amount   : RM 137.38
Net Salary     : RM 6809.88

Staff No       : 1007292
Salary         : RM 5400.00
EPF Amount     : RM 540.00
SOCSO Amount   : RM 121.50
Net Salary     : RM 4738.50
```

2. Create a text file called *student_file.txt*. The file has the following data which is the *student id* and *cgpa*.



```
1112222345 3.56
1111884352 1.20
1112543889 2.50
1113197710 3.90
1113193420 1.87
1112114001 1.55
```

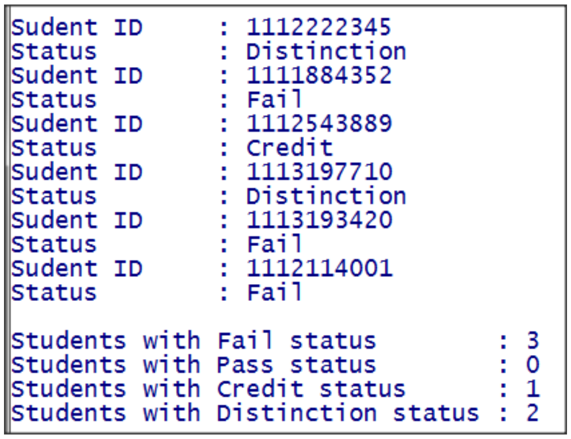
In the main() function:

- Create a FILE pointer called *fread*. Use this pointer to open the file *student_file.txt* for reading.
- If the file cannot be opened, display error message *"File cannot be accessed!"* and quit the program.
- Read all the data from the file.
- Call function *get_status()*, passing *cgpa* as parameter.
- Count how many students with *fail*, *pass*, *credit* and *distinction* status using if else statement.
- Display student id and status.
- Finally display the summary of results as shown below.

In function *get_status*():

– Using if else statement, identify the status of the cgpa and return it to main().

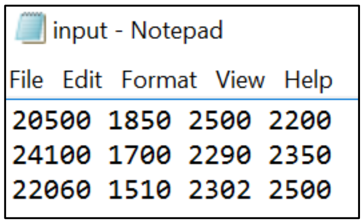
CGPA	Status
0.00 to less than 2.00	Fail
2.00 to less than 2.50	Pass
2.50 to less than 3.50	Credit
3.5 to 4.00	Distinction


```
Sudent ID      : 1112222345
Status         : Distinction
Sudent ID      : 1111884352
Status         : Fail
Sudent ID      : 1112543889
Status         : Credit
Sudent ID      : 1113197710
Status         : Distinction
Sudent ID      : 1113193420
Status         : Fail
Sudent ID      : 1112114001
Status         : Fail

Students with Fail status        : 3
Students with Pass status        : 0
Students with Credit status      : 1
Students with Distinction status : 2
```

3. You are required to write a program that calculates the total cost and also the cost per product for a company.

In the *main()* function :

- Create a file pointer called *fpinput*. Open the file input.txt for reading. The file contains the cost incurred by a company for 3 months which is the total staff *salary*, *maintenance* cost, *vehicle* cost and finally how many *units* of products that has been sold.



```
20500 1850 2500 2200
24100 1700 2290 2350
22060 1510 2302 2500
```

- If the file does not exist display "*The file could not be found*".
- Call function *get_total_cost(…)* and send the *salary*, *maintenance* and *vehicle* cost as parameters. This function will return the *total cost*.
- Call function *get_cost(…)* and send *total cost* and *units* sold as parameters. This function will return the cost to produce one product.
- Call function *store_record(…)* and send *total cost* and *cost per product* as parameters. This function will record all the cost and cost per product for the 3 months.
- Use if else statement to identify the highest total cost and the highest cost per product.
- Display the output as shown in the next page.
- Write the necessary prototypes for all functions.

In the *get_total_cost(…)* function:

- Calculate and return the total cost by adding up the *salary*, *maintenance* and *vehicle* costs.

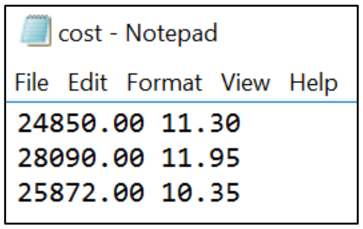
In the *get_cost(…)* function:

- Calculate and return the cost per unit of product by dividing the *total cost* with *units* sold.

In the *store_record(…)* function:

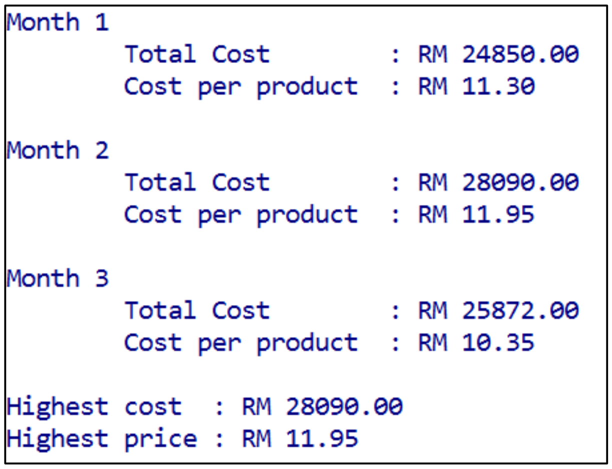
- Create a file pointer called *write*.
- Open a file called *cost.txt* for appending.
- Write the *total cost* and the *cost per product* into the file.

The sample file is shown below.



```
24850.00 11.30
28090.00 11.95
25872.00 10.35
```

Sample output:

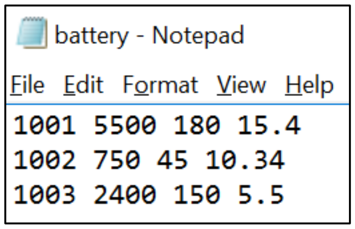


```
Month 1
        Total Cost          : RM 24850.00
        Cost per product    : RM 11.30

Month 2
        Total Cost          : RM 28090.00
        Cost per product    : RM 11.95

Month 3
        Total Cost          : RM 25872.00
        Cost per product    : RM 10.35

Highest cost  : RM 28090.00
Highest price : RM 11.95
```

4. You are required to write a complete program that calculates the life of a battery by using variables such as watt, voltage and resistance.

In the *main()* function :

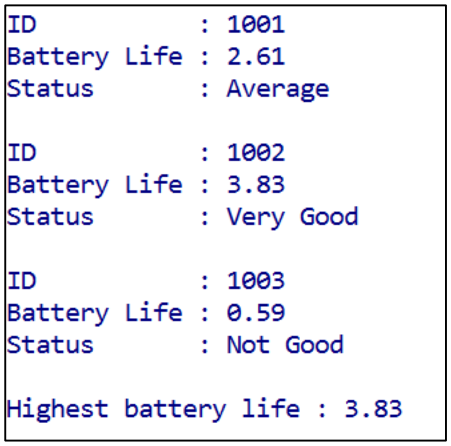
- Create a file pointer *finput* that reads the file called *battery.txt*. The contents of the file are *id, watt, voltage* and *resistance*. If the file does not exist, display a message "*File does not exist*".



```
1001 5500 180 15.4
1002 750 45 10.34
1003 2400 150 5.5
```

- Read all the 4 values from the file.
- Call function *get_current(…)* and pass the *voltage* and *resistance* of each battery.
- Call function *get_battery_life(…)* and pass the *watt, voltage* and *current*.
- Call function *get_status(…)* and pass the *battery life*.
- Call function store_record(…) and pass the *id* and *battery life*.
- Identify the highest battery life among all the batteries and display as shown below.

Sample output:



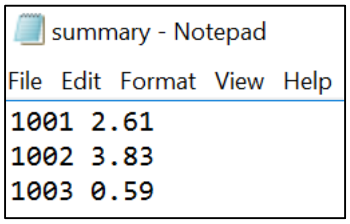
```
ID           : 1001
Battery Life : 2.61
Status       : Average

ID           : 1002
Battery Life : 3.83
Status       : Very Good

ID           : 1003
Battery Life : 0.59
Status       : Not Good

Highest battery life : 3.83
```

- In the *get_current(…)* function :
  - Calculate and return the *current* which can be calculated by the formula: $\frac{voltage}{resistance}$
- In the *get_battery_life(…)* function :
  - Calculate and return the *battery life* using the formula: $\frac{watt}{voltage\ x\ current}$
- In the *get_status(…)* function :
  - Identify and return the *status* of the battery by referring to the table given below.

Battery Life	Status
1.5 or less	Not Good
3 or less	Average
More than 3	Very Good

- In the *store_record(…)* function :
  - Using a file pointer called *fwrite*, open a file called *summary.txt* to append records.
  - Write the *id* and the *battery life* into the file as shown below.

Sample file content:



summary - Notepad

File Edit Format View Help

```
1001 2.61
1002 3.83
1003 0.59
```

5. Create a structure called Recycle with attributes *name* (string), *weight* and *income* (float).

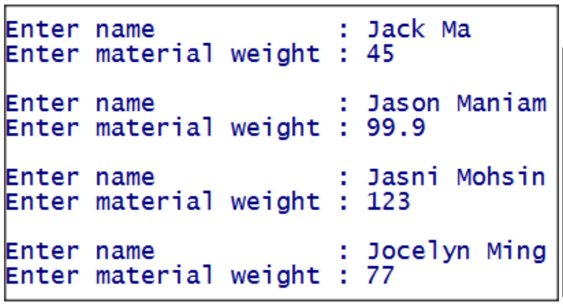
In the main() function:

- Create a structure variable array person with size 4.
- Create a FILE pointer called *fwrite*. Use this pointer to open the text file called recycle.txt for writing.
- Using for loop, ask the user to enter name and weight of recycle materials for 4 persons.
- Call function *get_price()*, passing *weight* as perimeter.
- Calculate the *income* for the recycled material (price x weight).
- Write the name, weight and income to the recycle.txt file.

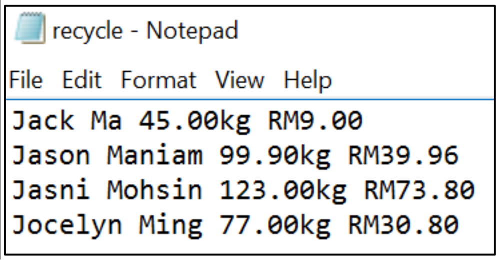
In function *get_price():*

- Based on the weight, identify the price and return it to main().
  - Weight less than 50kg, the price is RM 0.20 per kg.
  - Weight less than 100kg, the price is RM 0.40 per kg.
  - Weight more than 100kg, the price is RM 0.60 per kg.

The sample output and the content of recycle.txt after execution are shown below.



```
Enter name            : Jack Ma
Enter material weight : 45

Enter name            : Jason Maniam
Enter material weight : 99.9

Enter name            : Jasni Mohsin
Enter material weight : 123

Enter name            : Jocelyn Ming
Enter material weight : 77
```



recycle - Notepad

File Edit Format View Help

```
Jack Ma 45.00kg RM9.00
Jason Maniam 99.90kg RM39.96
Jasni Mohsin 123.00kg RM73.80
Jocelyn Ming 77.00kg RM30.80
```

6. Write a program that calculates the reward points given by a retailer for the purchase of products by its customers and update it to an existing file called *mesracard.txt*.

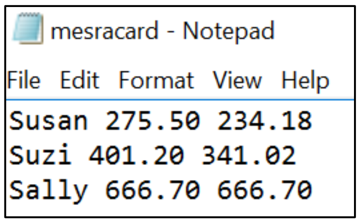
- Declare integer constant NUMBER using pre-processor directive and initialize to integer value 3.
- Create a structure called *Reward*. The data members are: *customerName (string), purchase* and *points (float).*

In *main( )* :
- Open a file called *mesracard.txt* for appending.
- If the file can't be opened, display error message File cannot be open. Program quitting and quit the program.
- Declare structure array called *record,* size NUMBER.
- Call function *calculate(…)*, passing in array *record*
- Using a *for-loop* :
  - Display each customer's name, purchase and points.
  - Write each customer's name, purchase and points into file mesracard.txt.
- Close file mesracard.txt.

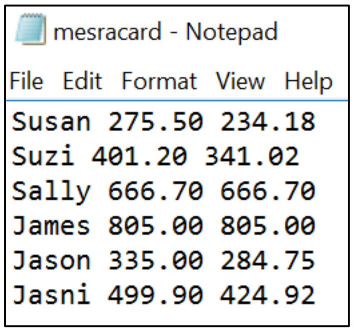
In function *calculate(...)*:
- Parameter : structure array record
- Return type : none
- Using a *for-loop* :
  - Get customer's name and purchase.
  - Use if-else statement to determine the points obtained based on the table below.

Total purchase (RM)	Reward Points
0-100	0.45 * purchase
100 < purchase <=250	0.7 * purchase
250 < purchase <=500	0.85 * purchase
More than 500	1 * purchase

The content of mesracard.txt before (left) and after (right) execution.

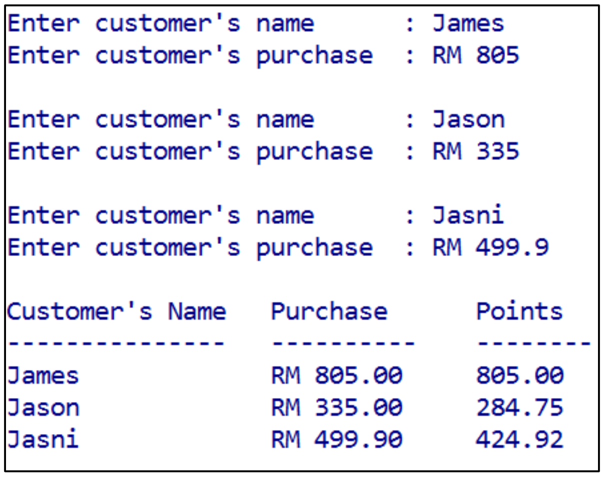


```
Susan 275.50 234.18
Suzi 401.20 341.02
Sally 666.70 666.70
```



```
Susan 275.50 234.18
Suzi 401.20 341.02
Sally 666.70 666.70
James 805.00 805.00
Jason 335.00 284.75
Jasni 499.90 424.92
```

Sample output:



```
Enter customer's name       : James
Enter customer's purchase   : RM 805

Enter customer's name       : Jason
Enter customer's purchase   : RM 335

Enter customer's name       : Jasni
Enter customer's purchase   : RM 499.9

Customer's Name    Purchase      Points
---------------    ----------    --------
James              RM 805.00     805.00
Jason              RM 335.00     284.75
Jasni              RM 499.90     424.92
```

7. Create a structure called Record with attributes *prodid, status* (string) and *rating* (array with size 3).

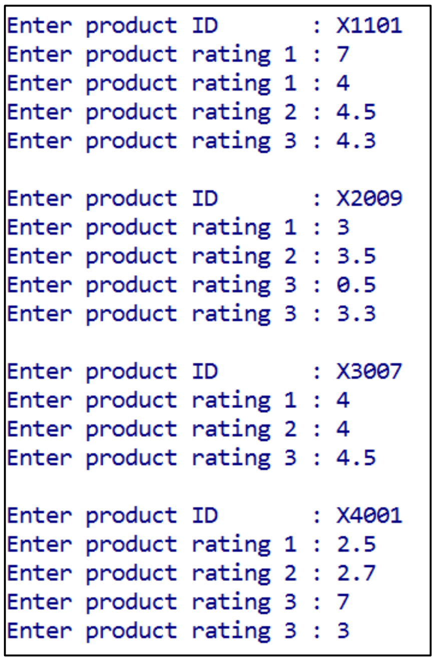
In the main() function:

- Create a structure variable array product with 4 elements.
- Create a FILE pointer called *fwrite*. Use this pointer to open the text file called product_rating.txt for writing.
- Using a while loop and repeat for 4 times, ask the user to enter product id.
  - Use for loop to get 3 ratings from the user.
  - The rating must be between 1 and 5. If the user enters an invalid rating value, ask the user to reenter the rating again.
  - Calculate total and average rating.
  - Call function *get_status(…)*, passing the average rating.
  - Write the product id, average rating and status to the product_rating.txt file.

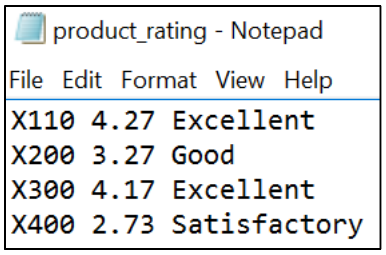
In function get_status(…):

- Based on the average, identify the status and return it to main().
  - Status “Poor” is for average lesser than 2.
  - Status “Satisfactory” is for average lesser than 3.
  - Status “Good” is for average lesser than 4.
  - Otherwise will be “Excellent”

The sample output and the content of product_rating.txt after execution are shown below.



```
Enter product ID        : X1101
Enter product rating 1 : 7
Enter product rating 1 : 4
Enter product rating 2 : 4.5
Enter product rating 3 : 4.3

Enter product ID        : X2009
Enter product rating 1 : 3
Enter product rating 2 : 3.5
Enter product rating 3 : 0.5
Enter product rating 3 : 3.3

Enter product ID        : X3007
Enter product rating 1 : 4
Enter product rating 2 : 4
Enter product rating 3 : 4.5

Enter product ID        : X4001
Enter product rating 1 : 2.5
Enter product rating 2 : 2.7
Enter product rating 3 : 7
Enter product rating 3 : 3
```



```
X110 4.27 Excellent
X200 3.27 Good
X300 4.17 Excellent
X400 2.73 Satisfactory
```

8. Write a program that creates a new file called *trainee.txt* in order to store the recent Microsoft Excel training details conducted at Multimedia University.

Create a structure called *Training*. The attributes are; *name* (string), *fee, cert, material, meal, cost, profit* (float). Declare structure variable array called *detail*, size 4.

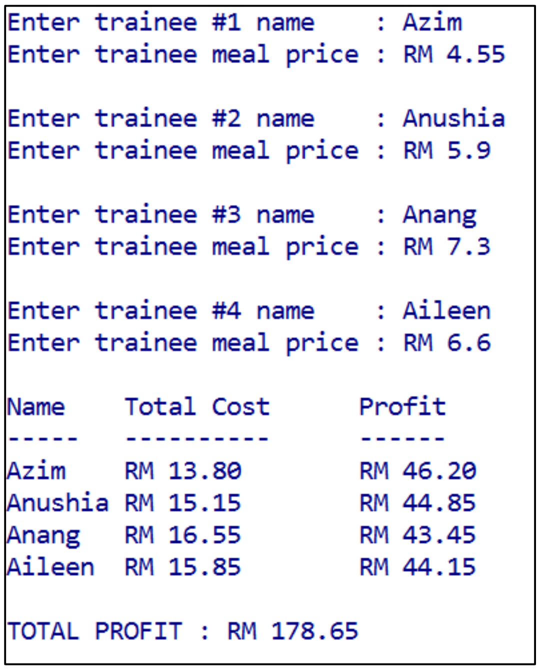
In *main( )* :

- Create a FILE pointer called *fptr*. Use this pointer to open the text file called trainee.txt for writing.
- Call function *get_total(…)*, passing in array *detail* as parameter (save return value into a variable).
- Using a *for-loop* and repeat for 4 times:
  - Display each trainee *name*, *cost* and *profit*.
  - Write each trainee *name*, *cost* and *profit* into file *trainee.txt*.
- Display the total profit returned from function *get_total (…)*.
- Close file *trainee.txt*.

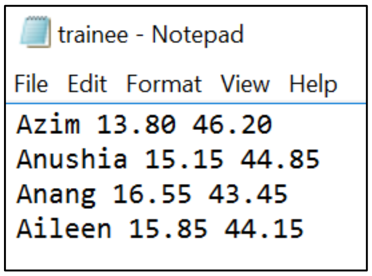
In function *get_total (...)*:

- Using a *for-loop* :
  - Get trainee *name* and *meal* price.
  - The training *fee* is fixed at RM60.
  - The cost to print each trainee *cert* and training *material* are RM1.30 and RM7.95 respectively.
  - Calculate each trainee total *cost* [*cert + material + meal*]
  - Calculate each trainee enrollment *profit* for the University [*fee – cost*]
  - Calculate the total profit of all 4 trainees.
- Return the total profit.

Sample output (left) and the *trainee.txt* content after execution:



```
Enter trainee #1 name     : Azim
Enter trainee meal price : RM 4.55

Enter trainee #2 name     : Anushia
Enter trainee meal price : RM 5.9

Enter trainee #3 name     : Anang
Enter trainee meal price : RM 7.3

Enter trainee #4 name     : Aileen
Enter trainee meal price : RM 6.6

Name     Total Cost       Profit
-----    ----------       ------
Azim     RM 13.80         RM 46.20
Anushia  RM 15.15         RM 44.85
Anang    RM 16.55         RM 43.45
Aileen   RM 15.85         RM 44.15

TOTAL PROFIT : RM 178.65
```



```
Azim 13.80 46.20
Anushia 15.15 44.85
Anang 16.55 43.45
Aileen 15.85 44.15
```

SUBMISSION

Write a program that calculates the total claims of the staffs in a company.

Create a structure called *Claims*. The structure includes attributes such as *name* (string), *mileage* (float), *days* (integer) and *claim* (float).

In the *main()* function:

- Create a structure variable array called *staff* which has 3 elements.
- Using a *do while loop*, ask the user to enter the *name*, *days* and *mileage*.
- Call function *get_mileage_amount(…)* and pass each staff's *mileage*.
- Call function *get_food_claim(…)* and pass each staff's *days*.
- Calculate each staff's claim by adding *mileage amount* and *food claim*.
- Call function *display_report(…)* and pass the structure variable *staff*.

In the *get_mileage_amount(…)* function:

- Set the *mileage rate* as constant using the **const** keyword to RM 0.70.
- Calculate and return the *mileage amount* based on the *mileage*. Each kilometer travelled costs RM 0.70.

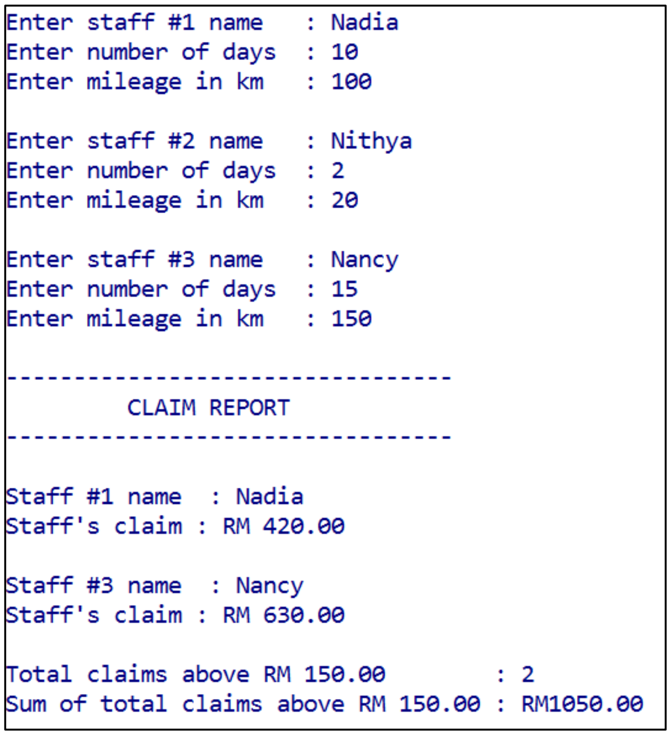
In the *get_food_claim(…)* function:

- Set the food rate as RM 35.00 using the **const** keyword.
- Calculate and return the amount that can be claimed based on the number of *days* worked. For each day, the staff can claim RM 35.00 for food.

In the *display_report(…)* function:

- Using ***for-loop***, identify and display which staff has claimed more than RM 150.00.
- Also display how many staff claimed more than RM 150.00 and the total sum.

Sample output:-



```
Enter staff #1 name   : Nadia
Enter number of days  : 10
Enter mileage in km   : 100

Enter staff #2 name   : Nithya
Enter number of days  : 2
Enter mileage in km   : 20

Enter staff #3 name   : Nancy
Enter number of days  : 15
Enter mileage in km   : 150

---------------------------------
         CLAIM REPORT
---------------------------------

Staff #1 name  : Nadia
Staff's claim : RM 420.00

Staff #3 name  : Nancy
Staff's claim : RM 630.00

Total claims above RM 150.00        : 2
Sum of total claims above RM 150.00 : RM1050.00
```