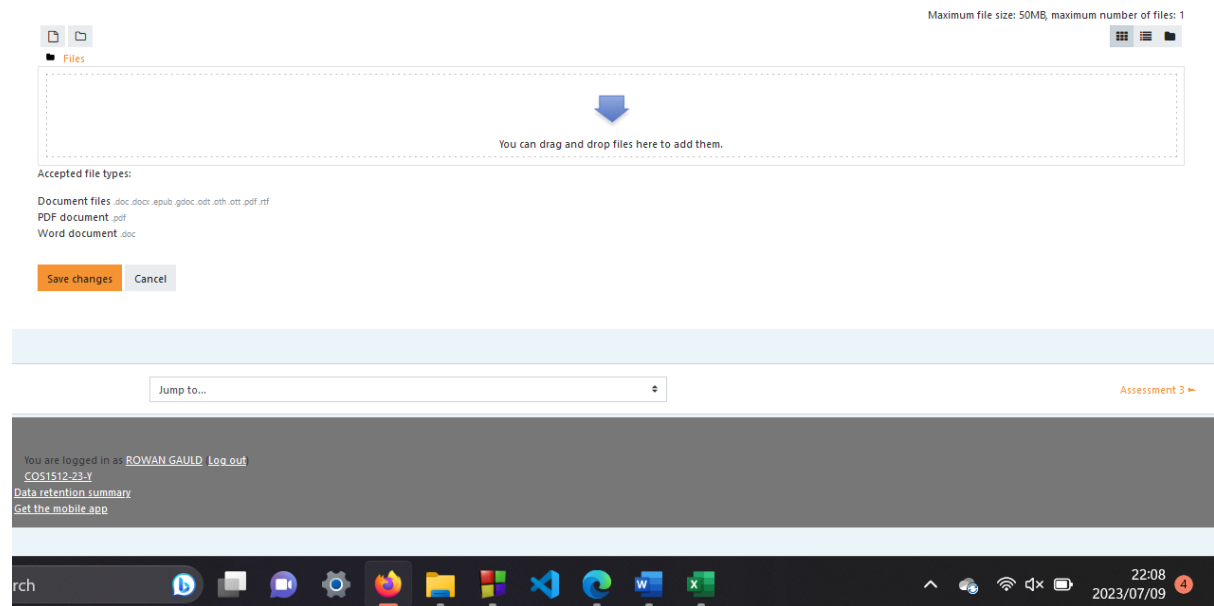


## Rowan Gauld 18339158 Assessment 2: 762913



### Question 1

Code:

```
#include <iostream>
using namespace std;
// Declaration - calcFees(int nModules, double moduleFees)
double calcFees(int nModules, double moduleFees);
// Declaration - calcFees(int nModules, double moduleFees, int nRepeatingModules, double repeatingModuleFees)
double calcFees(int nModules, double moduleFees, int nRepeatingModules, double repeatingModuleFees);
int main()
{
    bool repeatingModules;
    int nModules, nRepeatingModules;
    double moduleFees, repeatingModuleFees, totalFeesDue;
    // Prompting user to provide required information regardless if they are repeating or not
    cout << "Are you repeating any modules? Enter 1 for yes and 0 for no." << endl;
    cin >> repeatingModules;
    cout << "Please provide the number of modules you are taking for the first time/NOT repeating." << endl;
    cin >> nModules;
    cout << "Please provide the fees of your modules - they are the same price." << endl;
    cin >> moduleFees;
    if(repeatingModules)
    {
        // If the user is repeating, prompt them to provide the required information
        cout << "Please provide the number of modules you are repeating." << endl;
        cin >> nRepeatingModules;
        cout << "Please provide the fees of your repeating modules - they are the same price." << endl;
        cin >> repeatingModuleFees;

        // Calling - calcFees(int nModules, double moduleFees, int nRepeatingModules, double repeatingModuleFees)
        totalFeesDue = calcFees(nModules, moduleFees, nRepeatingModules, repeatingModuleFees);
    }
    else
    {
        // Calling - calcFees(int nModules, double moduleFees)
```

```

    totalFeesDue = calcFees(nModules, moduleFees);
}
// Displaying payable fees
cout << "The total amount of fees due by you are: R" << totalFeesDue;
return 0;
}
// Function - calcFees(int nModules, double moduleFees)
double calcFees(int nModules, double moduleFees)
{
    return nModules*moduleFees;
}
// Function - calcFees(int nModules, double moduleFees, int nRepeatingModules, double repeatingModuleFees)
double calcFees(int nModules, double moduleFees, int nRepeatingModules, double repeatingModuleFees)
{
    return (nModules*moduleFees) + (nRepeatingModules*repeatingModuleFees);
}

```

Output:

```

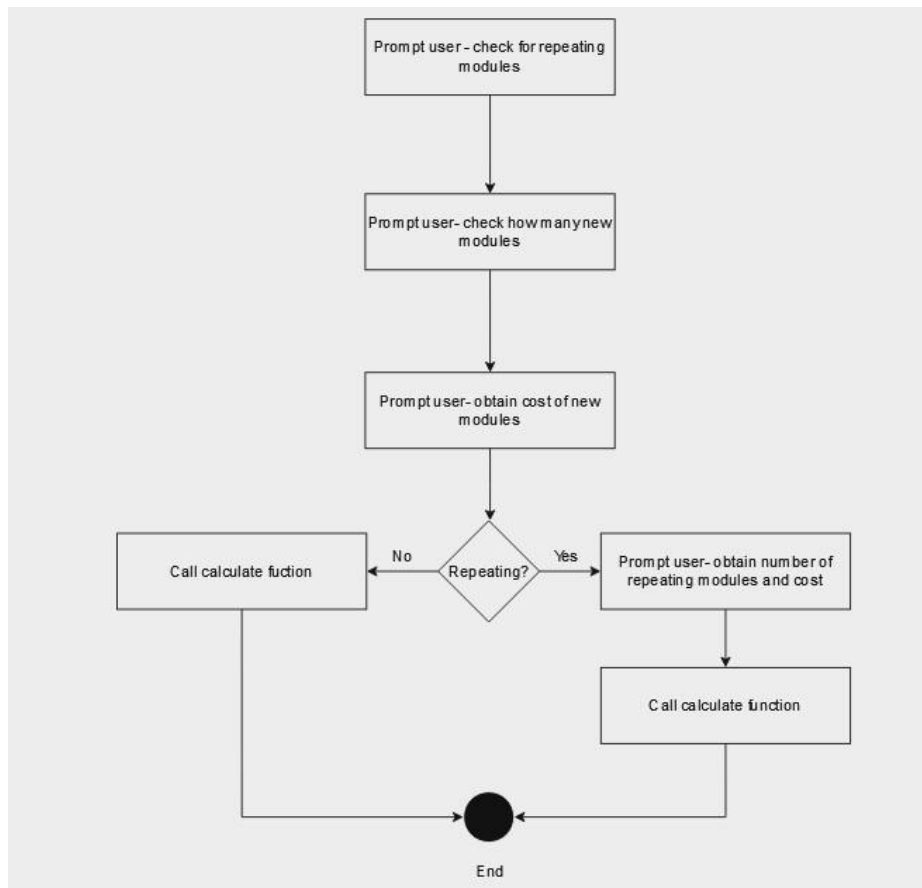
C:\Users\rowan\OneDrive\UN x + v
Are you repeating any modules? Enter 1 for yes and 0 for no.
0
Please provide the number of modules you are taking for the first time/NOT repeating.
3
Please provide the fees of your modules - they are the same price.
2000
The total amount of fees due by you are: R6000
Process returned 0 (0x0) execution time : 8.123 s
Press any key to continue.

```

```

C:\Users\rowan\OneDrive\UN x + v
Are you repeating any modules? Enter 1 for yes and 0 for no.
0
Please provide the number of modules you are taking for the first time/NOT repeating.
3
Please provide the fees of your modules - they are the same price.
2000
The total amount of fees due by you are: R6000
Process returned 0 (0x0) execution time : 8.123 s
Press any key to continue.

```



## Question 2

Code:

```

#include <iostream>
#include <string>
#include <cassert>
using namespace std;
int main()
{
    int hour, minute;
    string period;

    // Prompting user to provide required information
    cout << "Provide the hour: ";
    cin >> hour;
    // Asserting that the hour provided meets the 24 hour notation
    assert((hour >= 0) && (hour < 24));
    cout << "Provide the minute: ";
    cin >> minute;
    // Asserting that the minute provided meets the correct notation
    assert((minute >= 0) && (minute < 60));
    // Setting the correct time period
    if(hour < 12)
    {
        period = "AM";
    }
    else
    {
        period = "PM";
    }
}

```

```

// Mod 12 and the remainder is the correct hour
hour %= 12;
// Mod 00 or 0 does not work correctly, need to accommodate for that
if(hour == 0)
{
    hour = 12;
}
// Displaying the time in 12 hour notation
cout << "The time of day in 12-hour notation is: " << hour << ":" << minute << " " << period;
return 0;
}

```

Output:

```

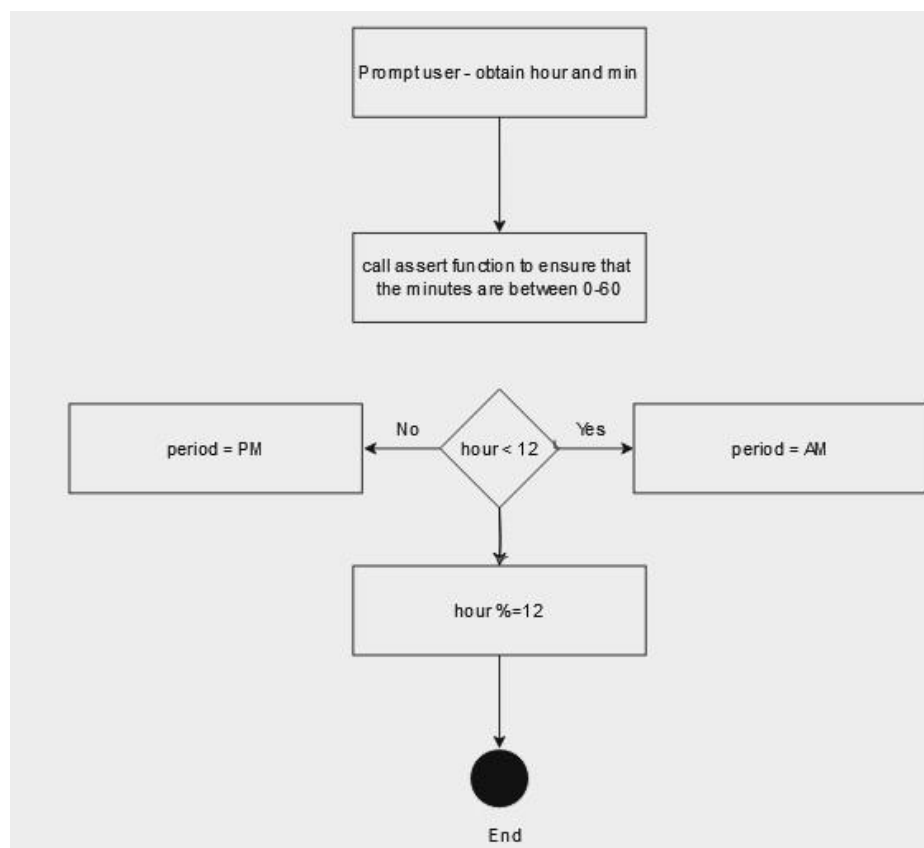
C:\Users\rowan\OneDrive\UN x + v
Provide the hour: 15
Provide the minute: 20
The time of day in 12-hour notation is: 3:20 PM
Process returned 0 (0x0)   execution time : 11.057 s
Press any key to continue.

```

```

C:\Users\rowan\OneDrive\UN x + v
Provide the hour: 10
Provide the minute: 22
The time of day in 12-hour notation is: 10:22 AM
Process returned 0 (0x0)   execution time : 5.171 s
Press any key to continue.

```



### Question 3

Right shift or get() ==> check for white space

Account No.	Opening Balance
€	Amount
C	Amount
€	Amount
C	Amount
C	Amount
C	Amount
C	Amount

int accountNo  
double openingBalance  
double amount  
char code

Account No.			
Opening Balance			
€	Amount	Balance	Bank Cost
C	Amount	Balance	Bank Cost
C	Amount	Balance	Bank Cost
C	Amount	Balance	Bank Cost
C	Amount	Balance	Bank Cost
C	Amount	Balance	Bank Cost
C	Amount	Balance	Bank Cost
Closing Balance			

string code  
double currentBalance  
double bankCost

## Question 4

Code:

```
#include <iostream>
#include <fstream>
using namespace std;
// Declaration - writeInputFile()
void writeInputFile();
int main()
{
    string inFile;
    string outFile;
    //Writing input to activity.dat
    writeInputFile();
    //Prompting the user to provide input file name
    cout << "Please enter name of input file: ";
    cin >> inFile;
    //Prompting the user to provide output file name
    cout << "Please enter name of output file: ";
    cin >> outFile;
    //Reading input file
    ifstream read(inFile);
    if (!read)
    {
        cout << inFile << "Does not exist...";
        exit(0);
    }
    //Reading each character and checking if it matches the code
    ofstream write(outFile);
    char ch;
    while (read.get(ch))
    {
        switch (ch)
        {
            case '0':
                ch = 's';
                break;

            case '1':
                ch = 'g';
                break;

            case '2':
                ch = 'o';
                break;

            case '3':
                ch = 'y';
                break;

            case '4':
                ch = 'v';
                break;

            case '5':
                ch = 'n';
                break;
```


```

    case '6':
        ch = 'f';
        break;

    case '7':
        ch = 'j';
        break;
    }
    //Reading output file
    write << ch;
}
return 0;
}
// Function - writelnputFile()
void writelnputFile()
{
    ofstream wrt("activity.dat");
    wrt << "We h2pe that 32u e5723ed the acti4it3. \n";
    wrt << "A6ter 32u ha4e c2mpleted the acti4it3, 0e5d 32ur re0ult t2: \n";
    wrt << "The Acti4it3 c2mpetiti25, Bett3 Da4i0 0treet 99, Auckla5d Park, \n";
    wrt << "989, a5d 0ta5d a cha5ce t2 wi5 a hamper c250i0ti51 26 c2l2uri51 \n";
    wrt << "a5d acti4it3 b22k0, c2l2uri51 pe5cil0 a5d pe50.";
    wrt.close();
}

```

Output:

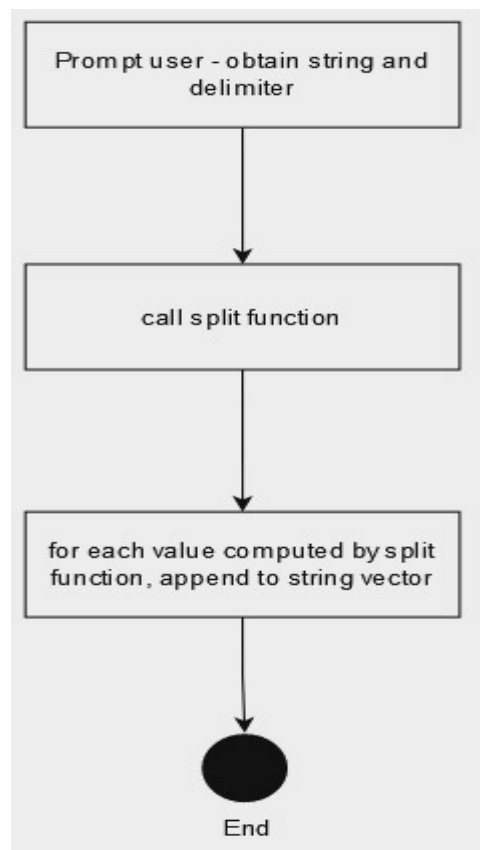


```

C:\Users\rowan\OneDrive\UN x + ~
Please enter name of input file: activity.dat
Please enter name of output file: competition.txt

Process returned 0 (0x0)   execution time : 10.038 s
Press any key to continue.

```



## Question 5

Code:

```
#include <iostream>
#include <cstring>
using namespace std;
// Declaration - void toLowerCase(char word[])
void toLowerCase(char word[]);
int main(void){
    char inputString[60];
    char words[10][20];
    int i,j=0,counter=0;
    // Prompting the user to provide a string
    cout << "Enter the person's name in the following format: first name, then middle name or initial, and then last
name:\n";
    gets(inputString);

    // For the length of the provided string
    for(i=0;i<=(strlen(inputString));i++){
        // Check if input string contains an ' ' or ending character - \0
        if(inputString[i]==' ' || inputString[i]=='\0'){
            words[counter][j]='\0';
            counter++;
            j=0;
        }
        else
        {
            words[counter][j]=inputString[i];
            j++;
        }
    }

    if(counter==3){
        toLowerCase(words[2]);
        toLowerCase(words[0]);
        words[1][0]=toupper(words[1][0]);
        printf("%s, %s %c.\n",words[2],words[0],words[1][0]);
    }else if(counter==2){
        toLowerCase(words[1]);
        toLowerCase(words[0]);
        printf("%s, %s\n",words[1],words[0]);
    }else if(counter==1){
        toLowerCase(words[0]);
        printf("%s\n",words[0]);
    }else{
        printf("Wrong input data.\n\n");
    }
    return 0;
}

// Function - void toLowerCase(char word[])
void toLowerCase(char word[]){
    word[0]=toupper(word[0]);
    for(int i=1; word[i]!='\0'; i++)
    {
        word[i]=tolower(word[i]);
    }
}
```



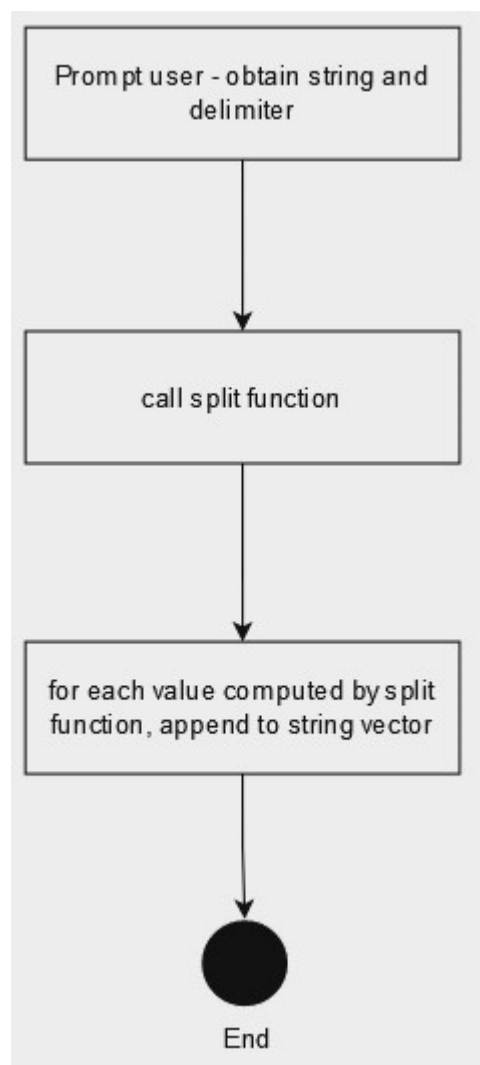
Output:

```
C:\Users\rowan\OneDrive\UN x + v
Enter the person's name in the following format: first name, then middle name or initial, and then last name:
Rowan Gauld
Gauld, Rowan

Process returned 0 (0x0)   execution time : 4.034 s
Press any key to continue.
```

```
C:\Users\rowan\OneDrive\UN x + v
Enter the person's name in the following format: first name, then middle name or initial, and then last name:
Rowan Middle Gauld
Gauld, Rowan M.

Process returned 0 (0x0)   execution time : 4.556 s
Press any key to continue.
```



## Question 6

Code:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
// Declaration - split(string sentence, char t)
vector<string> split(string sentence, char t);
int main()
{
    // Prompting the user to provide a string and a delimiter
```

```

        cout<<"Enter string : ";
        string sentence;
        cin>>sentence;
        char delimiter;
        cout<<"Enter delimiter : ";
        cin>>delimiter;
        vector<string>vectorSentence;

        // Calling - split(string s, char t)
        vectorSentence = split(sentence,delimiter);
        // Output each string produced by - split(string sentence, char t)
        for(int i=0; i<vectorSentence.size(); i++)
        {
            cout<<vectorSentence[i]<<" ";
        }
        return 0;
}
// Function -- split(string sentence, char t)
vector<string> split(string sentence, char delimiter)
{
    vector<string>vectorSentence;
    string str="";
    for(int i=0; i<sentence.length(); i++)
    {
        // If character is the delimiter
        if(sentence[i] == delimiter)
        {
            //Append empty string
            vectorSentence.push_back(str);
            str="";
        }
        else{
            // Concat character to string
            str += sentence[i];
        }
    }
    // Append the string
    vectorSentence.push_back(str);
    return vectorSentence;
}

```

Output:

```

C:\Users\rowan\OneDrive\UN >
Enter string : do, re, me, fa, so, la, ti, do
Enter delimiter : ,
do re me fa so la ti do
Process returned 0 (0x0)   execution time : 18.405 s
Press any key to continue.

```

## Question 7

- What is a pointer? A pointer is a construct that gives you more control of the computer's memory, it is the memory address of a variable.
- What is a dereferencing operator? The dereferencing operator is \*, it produces the variable it points to.
- What is the difference between assignment statements  $p1 = p2$ ; and  $*p1 = *p2$ ? The first is setting the value stored in variable p2 to p1. The second statement, the memory address of p2 is stored in p1.

- D. What is a dynamic variable? It returns a pointer that points to a new variable – they are created and destroyed while the program is running.
- E. What is the purpose of the new operator? It creates a new dynamic variable of a specified type and returns a pointer that points to the new variable.
- F. What is the purpose of the delete operator? Eliminates a dynamic variable that returns the memory that the dynamic variable occupied to the freestore. The memory can then be reused.
- G. What is the freestore (also called the heap)? A type of memory that is reserved for dynamic variables.
- H. What is the difference between dynamic variables and automatic variables? Dynamic variables are created and destroyed while the program is running whereas automatic variables have their dynamic properties managed by the computer – when the function that calls them is called and finishes.
- I. What is a dynamic array? An array whose size is not specified when the program is written but determined while the program is running.
- J. What is the advantage of using dynamic arrays? If you do not know how large your array needs to be, it can provide more flexibility.
- K. What is the relationship between pointers and arrays? An array is represented by a variable that is associated with the address of its first storage location whereas a pointer is also the address of a storage location with a defined type.
- L. Write statements to do the following: i.
  - a. Define a pointer type `int_ptr` for pointer variables that contain pointers to int variables.
  - b. Declare `p1` to be a pointer to an int.
  - c. Dynamically allocate an integer variable and store its address in `p1`. iv. Assign the value 23 to the variable that `p1` is pointing to. v. Declare an int variable `a`. vi. Let `p1` point to `a`.
  - d. Free the memory allocated to the variable that `p1` is pointing to.
- J. Write statements to do the following:
  - a. Define a pointer type `int_ptr` for pointer variables that contain pointers to int variables.
  - b. Declare `p2` to be a pointer to an int.
  - c. Obtain an integer value `nrElements` from the user indicating the number of elements to allocate.
  - d. Dynamically allocate an array of `nrElements` integers and store its address in `p2`.
  - e. Declare an int array `a` with 500 elements.
  - f. Assume `p2` has been initialized and copy the elements of `p2` one by one to the corresponding elements in `a`.
  - g. Free the memory allocated to the variable that `p2` is pointing to.
- K. Write a program that asks a user to enter the size of a dynamic array that stores exam marks obtained by students. Create the dynamic array and a loop that allows the user to enter an exam mark into each array element. Loop through the array, find the average mark for the exam and output it. Delete the memory allocated to your dynamic array before exiting your program.