Programing Tasks Report Content

1. Command Line Calculator
    A. Command Line Calculator for individual integer numbers.(simplecalc) ………..p2
    B. Command Line Calculator for Fields of integer numbers.(veccalc) ………………p4

2. Parking Fees(parkingfees)…………… p6

3. Detecting and discarding duplicate Entries(simplestats) ……………p8

4. Guess-a-Number Game(numberguess) …………..p10

5. Appendix(C++ code)……………p13

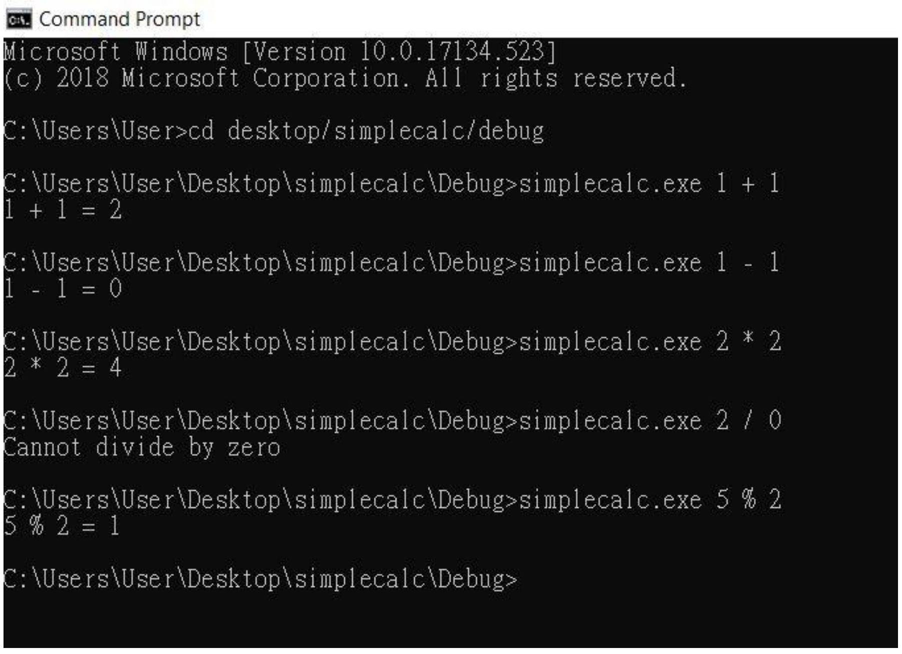
1. Command Line Calculator

A. Command Line Calculator for individual integer numbers. (simplecalc)

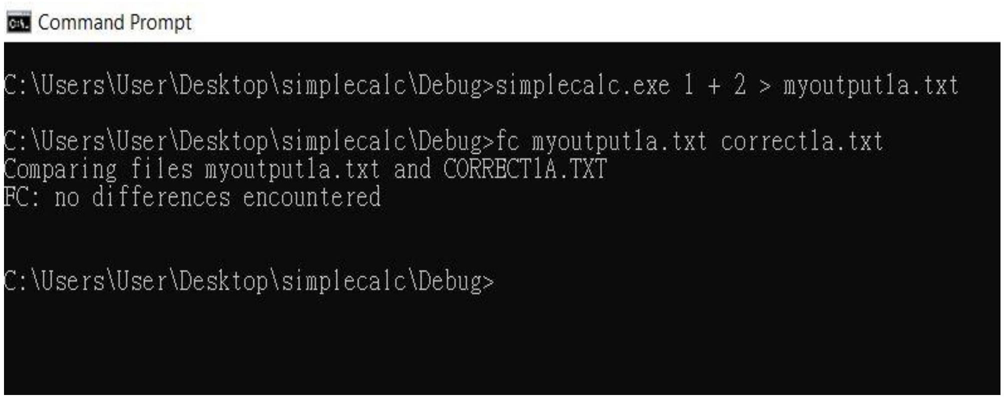
(1) Problem description

This is about creating a command line calculator. A first number followed by an operator followed by a second number is passed via command line. Depending on the operator, the numbers should be calculated accordingly. The operations addition, subtraction, multiplication, division, and modulo for integers are to be realized with the functions `add(…)`, `sub(…)`, `mul(…)`, `div(…)`, and `mod(…)`, respectively. Here and below, the three dots in parentheses after a function name are intended to indicate that the function expects arguments. These functions accept a first argument and a second argument, both of type `int`, via call-by-value. They should also accept a third argument of type `bool` by means of call-by-reference. The functions should return the result of the calculation as an integer (type `int`). If in the function `div(…)` or in the function `mod(…)` the second number is zero, the `bool` variable should be set to `false` and the first number is to be returned as result in this case. In all other cases the `bool` variable must be set to `true`.

(2) Screen shots



```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>cd desktop/simplecalc/debug

C:\Users\User\Desktop\simplecalc\Debug>simplecalc.exe 1 + 1
1 + 1 = 2

C:\Users\User\Desktop\simplecalc\Debug>simplecalc.exe 1 - 1
1 - 1 = 0

C:\Users\User\Desktop\simplecalc\Debug>simplecalc.exe 2 * 2
2 * 2 = 4

C:\Users\User\Desktop\simplecalc\Debug>simplecalc.exe 2 / 0
Cannot divide by zero

C:\Users\User\Desktop\simplecalc\Debug>simplecalc.exe 5 % 2
5 % 2 = 1

C:\Users\User\Desktop\simplecalc\Debug>
```



```
C:\Users\User\Desktop\simplecalc\Debug>simplecalc.exe 1 + 2 > myoutput1a.txt

C:\Users\User\Desktop\simplecalc\Debug>fc myoutput1a.txt correct1a.txt
Comparing files myoutput1a.txt and CORRECT1A.TXT
FC: no differences encountered


C:\Users\User\Desktop\simplecalc\Debug>
```

(3)Short review

In this task, I learned that by adding int main(int argc, char* argv[]), we are able to pass value from command line. I also learned more skills to use command line more efficiently such as pressing tab and up and down arrow key. Passing bool using pass by reference make me understand that this allow the value to pass in two-way, compare to normal value passing, which is one-way.

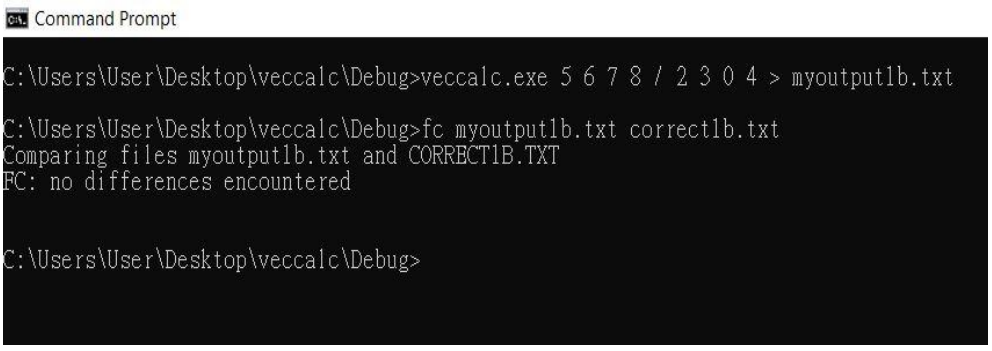
(4) Code $\longrightarrow$ See appendix 1A.

B. Command Line Calculator for Fields of integer numbers.(veccalc)

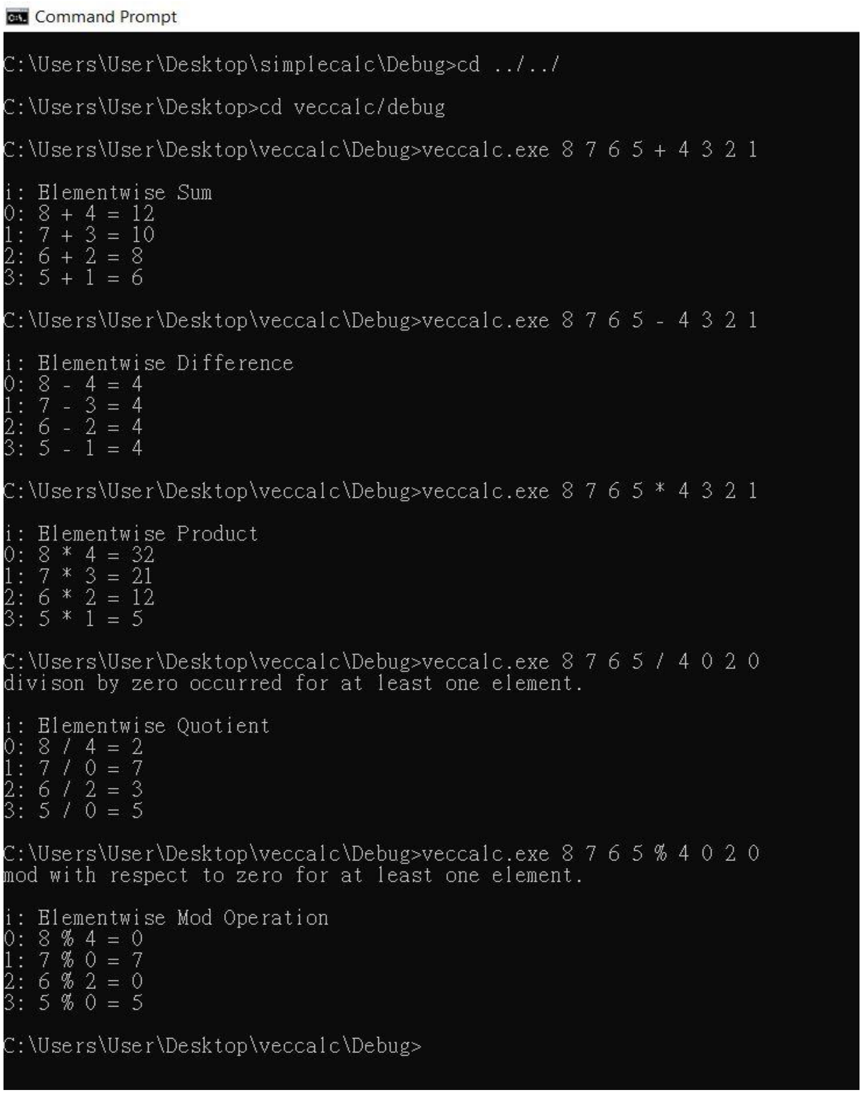
(1) Problem description

It is an extended version of *simplecalc* that allows the processing of fields represented as `vector<int>` in C++. The basic procedure is the same as for *simplecalc*. Only now two fields with integers are passed. They have to be processed element by element. Analogous to *simplecalc*, the operations addition, subtraction, multiplication, division, and modulo are to be implemented. Again, the functions should be called `add(…)`, `sub(…)`, `mul(…)`, `div(…)`, and `mod(…)`, but this time they should operate on fields of type `vector<int>`. Here and below, the three dots in parentheses after a function name are intended to indicate that the function expects arguments. If a zero is found in the second field during division or modulo calculation, a `bool` variable must be set to `false`. In this case, the corresponding number from the first field must be returned as the result of the operation. Since the solution of *veccalc* is again a field, a corresponding output function is needed. We implement a suitable `display(…)` function.

(2) Screen shots



```
Command Prompt

C:\Users\User\Desktop\veccalc\Debug>veccalc.exe 5 6 7 8 / 2 3 0 4 > myoutput1b.txt

C:\Users\User\Desktop\veccalc\Debug>fc myoutput1b.txt correct1b.txt
Comparing files myoutput1b.txt and CORRECT1B.TXT
FC: no differences encountered


C:\Users\User\Desktop\veccalc\Debug>
```


Command Prompt

```
C:\Users\User\Desktop\simplecalc\Debug>cd ../../

C:\Users\User\Desktop>cd veccalc/debug

C:\Users\User\Desktop\veccalc\Debug>veccalc.exe 8 7 6 5 + 4 3 2 1

i: Elementwise Sum
0: 8 + 4 = 12
1: 7 + 3 = 10
2: 6 + 2 = 8
3: 5 + 1 = 6

C:\Users\User\Desktop\veccalc\Debug>veccalc.exe 8 7 6 5 - 4 3 2 1

i: Elementwise Difference
0: 8 - 4 = 4
1: 7 - 3 = 4
2: 6 - 2 = 4
3: 5 - 1 = 4

C:\Users\User\Desktop\veccalc\Debug>veccalc.exe 8 7 6 5 * 4 3 2 1

i: Elementwise Product
0: 8 * 4 = 32
1: 7 * 3 = 21
2: 6 * 2 = 12
3: 5 * 1 = 5

C:\Users\User\Desktop\veccalc\Debug>veccalc.exe 8 7 6 5 / 4 0 2 0
divison by zero occurred for at least one element.

i: Elementwise Quotient
0: 8 / 4 = 2
1: 7 / 0 = 7
2: 6 / 2 = 3
3: 5 / 0 = 5

C:\Users\User\Desktop\veccalc\Debug>veccalc.exe 8 7 6 5 % 4 0 2 0
mod with respect to zero for at least one element.

i: Elementwise Mod Operation
0: 8 % 4 = 0
1: 7 % 0 = 7
2: 6 % 2 = 0
3: 5 % 0 = 5

C:\Users\User\Desktop\veccalc\Debug>
```

(3)Short review

In this task, I got more familiar to the use of vector. However, I spent a lot of time trying to figure out how to get the symbol that is in char* argv[] and store it into a char variable. I also had a hard time to determine the range of the for loop so I can store the element into the vector correctly. The abort() and vector subscript out of range errors occur many times when I was debugging.

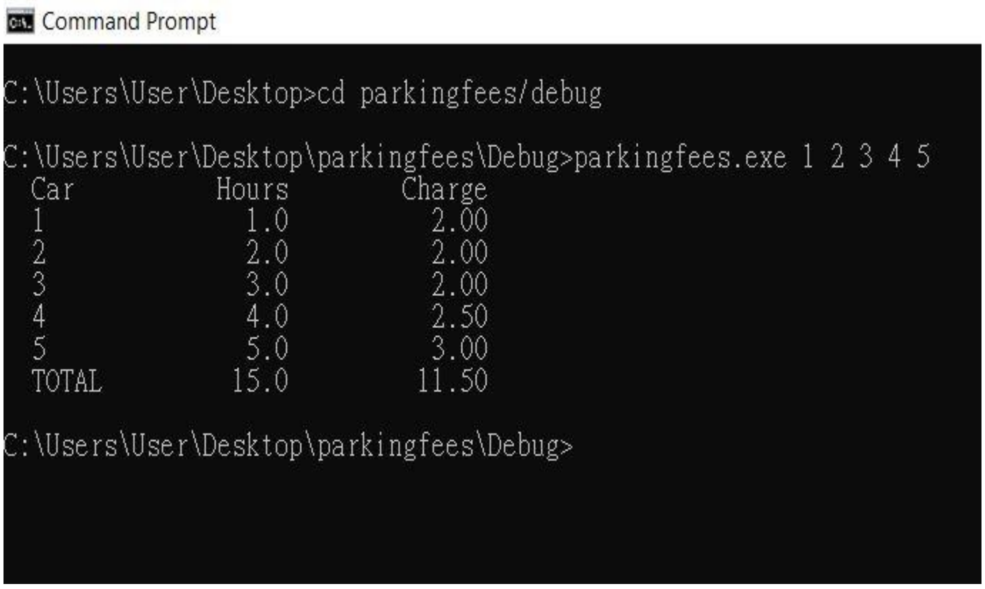
(4)Code ⟹ See appendix 1B.

2. Parking Fees(parkingfees)

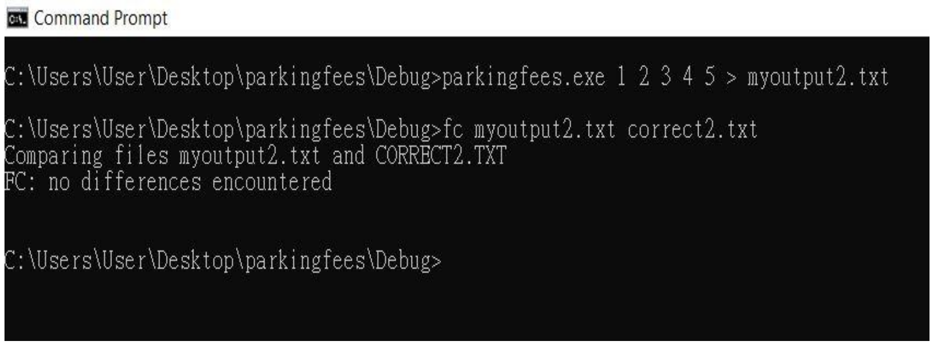
(1)Problem description

A parking garage charges a EUR 2.00 minimum fee to park for up to three hours. The garage charges an additional EUR 0.50 per hour after the first three hours. The maximum charge for any given 24-hour period is EUR 10.00. Assume that no car parks for longer than 24 hours at a time. Write a program *parkingfees.exe* that takes the numbers of hours as a command line argument. To provide more functionality, it should also be possible to enter the hours of multiple customers on the command line as well. The program should print the results in a neat tabular format. There, it should list the hours, the associated fees, the total of the hours and the total of the payments. We represent the numbers using the data type `double`.

(2)Screen shots



```
C:\Users\User\Desktop>cd parkingfees/debug

C:\Users\User\Desktop\parkingfees\Debug>parkingfees.exe 1 2 3 4 5
  Car          Hours        Charge
  1              1.0          2.00
  2              2.0          2.00
  3              3.0          2.00
  4              4.0          2.50
  5              5.0          3.00
  TOTAL         15.0         11.50

C:\Users\User\Desktop\parkingfees\Debug>
```



```
C:\Users\User\Desktop\parkingfees\Debug>parkingfees.exe 1 2 3 4 5 > myoutput2.txt

C:\Users\User\Desktop\parkingfees\Debug>fc myoutput2.txt correct2.txt
Comparing files myoutput2.txt and CORRECT2.TXT
FC: no differences encountered


C:\Users\User\Desktop\parkingfees\Debug>
```

(3)Short review

I learn some new statements such as setw(), setprecision(), in this tasks. These statements allow us to arrange the output instead of endl, ‘\n’, ’\t’, ” ” etc.

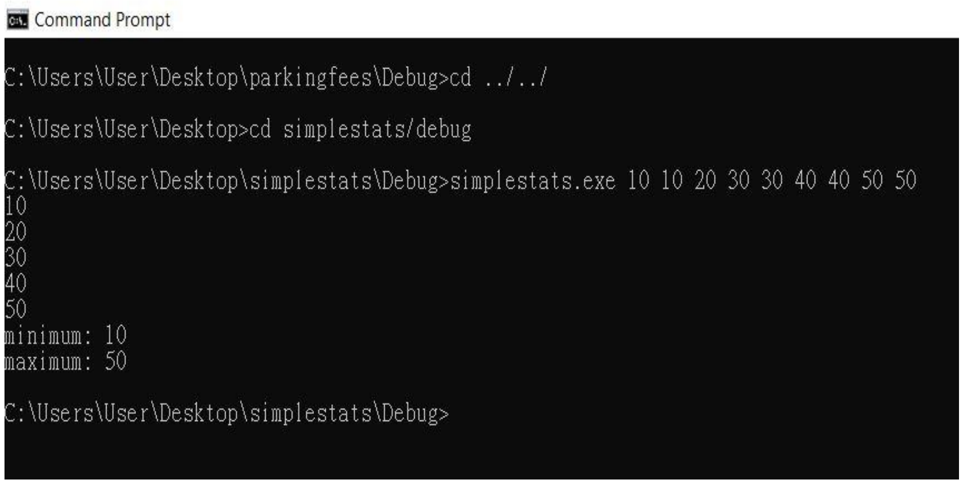
(4)Code $\longrightarrow$ See appendix 2.

3. Detecting and discarding duplicate Entries (simplestats)

(1) Problem description

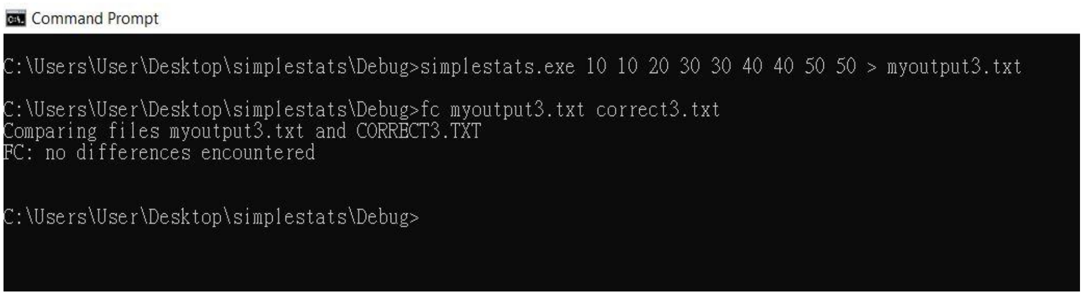
Use a one-dimensional vector to solve the following problem. Read in numbers from the command line. Store each of the input numbers in a vector (of type `int`), but only if the number is between 10 and 100 (10 <= number <= 100), and only if it isn't a duplicate of a number already read. After processing all the input values this way, display the unique values that the user entered as shown below. Then calculate the minimum and the maximum of the (unique) numbers entered using the functions `min(…)` and `max(…)`, respectively, and show them.

(2) Screen shots



Command Prompt

```
C:\Users\User\Desktop\parkingfees\Debug>cd ../../

C:\Users\User\Desktop>cd simplestats/debug

C:\Users\User\Desktop\simplestats\Debug>simplestats.exe 10 10 20 30 30 40 40 50 50
10
20
30
40
50
minimum: 10
maximum: 50

C:\Users\User\Desktop\simplestats\Debug>
```



Command Prompt

```
C:\Users\User\Desktop\simplestats\Debug>simplestats.exe 10 10 20 30 30 40 40 50 50 > myoutput3.txt

C:\Users\User\Desktop\simplestats\Debug>fc myoutput3.txt correct3.txt
Comparing files myoutput3.txt and CORRECT3.TXT
FC: no differences encountered


C:\Users\User\Desktop\simplestats\Debug>
```

(3) Short review

The two main challenges in this task are, first, to check whether the element already occurred before. If it did, we don’t save the element again into the vector. I figured out that I can use a loop checking backward and a bool value to get over this.

(4) Code ➡ See appendix 3.

4. Guess-a-Number Game(numberguess)

(1) Problem description

In this task, an integer number is to be guessed. At first, a number between 1 and 100 is to be entered. Then the guesser has to find the number. If his answer is wrong, he gets the hint that his number was either too low or too high. As a default setting, the guesser has three attempts to find the correct number. However, it should be possible to enter a different number of attempts (as low as one and as high as seven). To this end, the player should be asked at the beginning of the game how many attempts are desired. The following functions should be implemented completely.

a) `bool checkNumber(int number):`
The number to be guessed is passed to this function. If the number is between 1 (= LowerLimitNumber) and 100 (= UpperLimitNumber), the function returns `true` otherwise `false`. Please use two constant global variables for the numbers. Name the numbers `UpperLimitNumber` and `LowerLimitNumber`.

`b) bool checkTries(int tries):`
The number of allowed attempts is to be passed to this function. If the number is between 3 (=`LowerLimitTries`) and 7 (= `UpperLimitTries`), then the function returns `true`, otherwise `false`. Please use two global constant variables for `UpperLimitTries` and `LowerLimitTries`.

`c) bool setupGame(int& numberRef, int& triesRef):`
A reference, `numberRef`, to the variable `number` and a reference, `triesRef`, to the variable `tries` are passed to this function. Within this function, the user should be asked to enter the number to be guessed until he has entered a valid number. The same applies to the number of attempts. If the function is executed successfully, it returns `true`. If the user has created "`eof`" by entering Ctrl+Z instead of a regular input, the function returns `false`. If Ctrl+Z is entered for the number, the message "`You do not want to set a number, so you stopped the program`" should be output. If Ctrl+Z was entered for the number of tries, the function should print "`You do not want to set`

`tries, so you stopped the program".` The function `setupGame(…)` should use the functions `checkNumber(…)` and `checkTries(…)` as subfunctions.

`d) bool guessNumber(int guess, int correct, int& triesRef):`

In this function the number is to be guessed. The function should return `false` if the number, `guess`, does not match `correct`. If the number `guess` is identical with the number `correct`, the function returns `true` and the expression `"With the number" guess "you guessed the right number and you still had" tries " tries remaining".` Each time the function is called, the `tries` variable, which is passed by reference, `triesRef`, should be lowered by one. If this variable has a value smaller than 1, the function also returns `false`. In addition, the console window should show whether the guessed number was too big or too small. In the first case, the function should print "`The number you guessed was too big`". In the second case, it should print "`The number you guessed was too small`".

`e) int makeAGuess(int correctnumber, int& triesRef):`

In this function, the user is asked to enter a number. If the user has entered the value "`eof`" using Ctrl+Z, the function makeAGuess(…) returns -1. A frustrated user can enter this to end the current game. Otherwise the number entered in this function together with `correctnumber` and `triesRef` will be passed on to the `guessNumber(…)` function. The return values of makeAGuess are as follows: the function `makeAGuess(…)` should return -1, if the user has entered Ctrl+Z, it should return 1 if `guessNumber(…)` returns `true`, and it should return 0 if `guessNumber(…)` returns `false`.

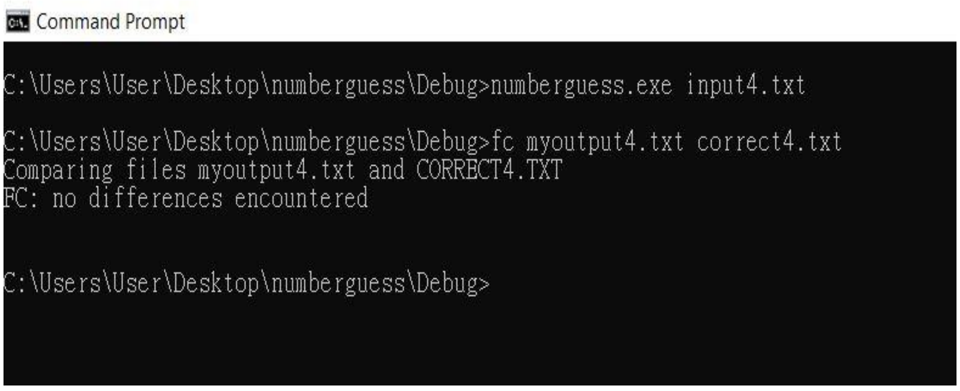
f) `int playGame():`

This function starts the game. It uses the `setupGame(…)` function as subfunction to specify a guess number and the number of attempts. If `setupGame(…)` returns `false`, `playGame(…)` must return -1 and print "You do not want to play the game". If `setupGame(…)` returns `true`, the user should then be able to guess the number in the `makeAGuess(…)` function and there in the `guessNumber()` subfunction. If `makeAGuess(…)` returns -1, then `playGame(…)` should also end with -1 and the text `"You do not want to play the game"` should be displayed. The function `playGame(…)` returns 1, if the number was guessed in time. If the number wasn't guessed in time, it returns 0 and prints out the text `"You ran out of tries"`. After each guessing attempt, the remaining number of attempts should be communicated to the user via the following text: `"You have " tries "tries remaining"`.

g) Now call the function `playGame()` within your `main()` function.

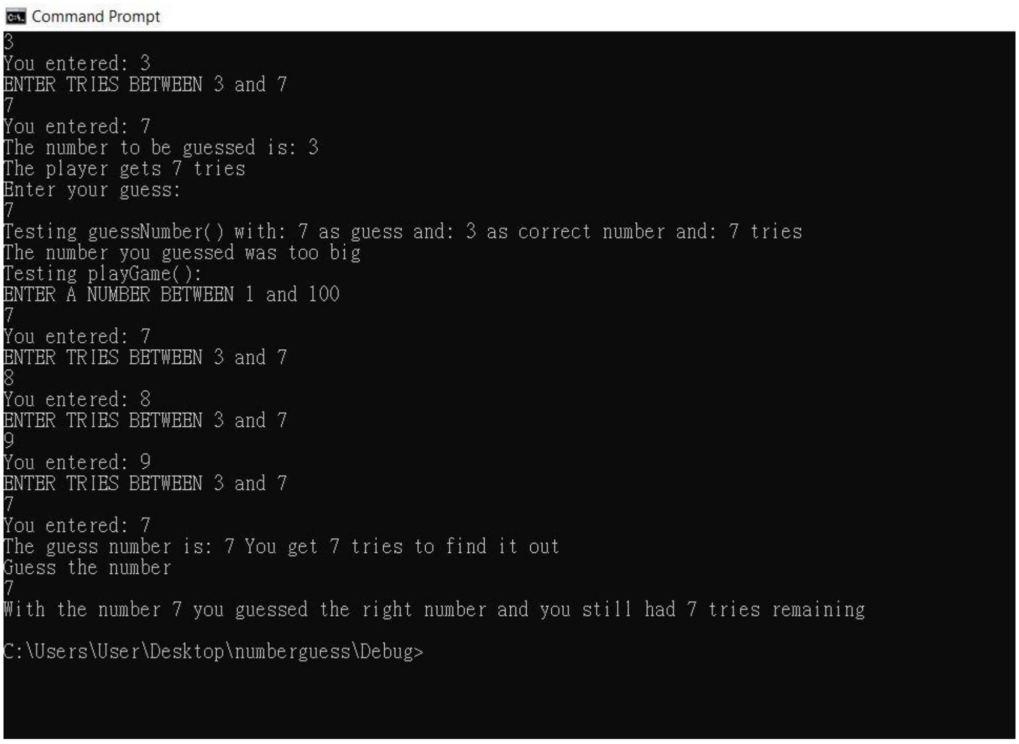
h) As a last step, check whether your number guessing game works as intented by using the supplied input4.txt file. Your program should create a file myoutput4.txt, the content of which should be identical to the content of the downloaded output4.txt file. Use "fc myoutput4.txt output4.txt" to compare your result with output4.txt.

(2) Screen shots



Command Prompt

```
C:\Users\User\Desktop\numberguess\Debug>numberguess.exe input4.txt

C:\Users\User\Desktop\numberguess\Debug>fc myoutput4.txt correct4.txt
Comparing files myoutput4.txt and CORRECT4.TXT
FC: no differences encountered


C:\Users\User\Desktop\numberguess\Debug>
```



```
3
You entered: 3
ENTER TRIES BETWEEN 3 and 7
7
You entered: 7
The number to be guessed is: 3
The player gets 7 tries
Enter your guess:
7
Testing guessNumber() with: 7 as guess and: 3 as correct number and: 7 tries
The number you guessed was too big
Testing playGame():
ENTER A NUMBER BETWEEN 1 and 100
7
You entered: 7
ENTER TRIES BETWEEN 3 and 7
8
You entered: 8
ENTER TRIES BETWEEN 3 and 7
9
You entered: 9
ENTER TRIES BETWEEN 3 and 7
7
You entered: 7
The guess number is: 7 You get 7 tries to find it out
Guess the number
7
With the number 7 you guessed the right number and you still had 7 tries remaining

C:\Users\User\Desktop\numberguess\Debug>
```

(3) Short review

The most challenging part of this task is there are many subfunctions, and they are all passing values and references. It’s so hard to understand the logic and structure of this program. Sometime each subfunction share the same variable name but sometime they don’t, which make me even harder to specify whether they have the same value or not.

(4) Code ⟹ See appendix 4.

5. Appendix(C++ code)

1A:

```
/****************************
*Filename:simplecalc
*Date:29.12.2018
*Author:Chuang, Tzu-Yi
*matriculation number:4018095
****************************/
#include<iostream>
#include<string>
using namespace std;
int add(int a, int b, bool &i)//add function
{
     int ans;
     ans = a + b;
     return ans;
}
int sub(int a, int b, bool &i)//sub functiion
{
     int ans;
     ans = a - b;
     return ans;
}
int mul(int a, int b, bool &i)//mul function
{
     int ans;
     ans = a * b;
     return ans;
}
int div(int a, int b, bool &i)//div function
{
     int ans;
     if (b == 0)
     {
          i = 0;
          return a;
     }//If the second number is 0, return the first number.
```

```
    else
    {
        i = 1;
        ans = a / b;
        return ans;
    }
}
int mod(int a, int b, bool &i)//mod function
{
    int ans;
    if (b == 0)
    {
        i = 0;
        return a;
    }//If the second number is 0, return the first number.
    else
    {
        i = 1;
        ans = a % b;
        return ans;
    }
}
int main(int argc, char *argv[])
{
    int a = stoi(argv[1]), b = stoi(argv[3]), ans;
    bool i = 1;
    char t = argv[2][0];
    if (t == '+')
    {
        ans = add(a, b, i);
        cout << a << " " << t << " " << b << " = " << ans << endl;
    }
    else if (t == '-')
    {
        ans = sub(a, b, i);
        cout << a << " " << t << " " << b << " = " << ans << endl;
    }
    else if (t == '*')
```

```
        {
            ans = mul(a, b, i);
            cout << a << " " << t << " " << b << " = " << ans << endl;
        }
        else if (t == '/')
        {
            ans = div(a, b, i);
            if (i == 0)//bool value has been passed by reference,
                       //hence is changed in both main function and div function.
            {
                cout << "Cannot divide by zero" << endl;
            }
            else
            {
                cout << a << " " << t << " " << b << " = " << ans << endl;
            }
        }
        else if (t == '%')
        {
            ans = mod(a, b, i);
            if (i == 0)//bool value has been passed by reference,
                       //hence is changed in both main function and mod function.
            {
                cout << "Cannot calculate modulo with respect to 0" << endl;
            }
            else
            {
                cout << a << " " << t << " " << b << " = " << ans << endl;
            }
        }
        return 0;
    }
```

1B:

```
/****************************
*Filename:veccalc
*Date:30.12.2018
*Author:Chuang, Tzu-Yi
*matriculation number:4018095
****************************/
#include<iostream>
#include<vector>
#include<string>
using namespace std;
vector<int> add(vector<int> a, vector<int> b, bool &status)//add function
{
     int i;
     vector<int> c;
     for (i = 0; i < a.size(); i++)
     {
          c.push_back(a[i] + b[i]);
     }
     return c;
}

vector<int> sub(vector<int> a, vector<int> b, bool &status)//sub function
{
     int i;
     vector<int> c;
     for (i = 0; i < a.size(); i++)
     {
          c.push_back(a[i] - b[i]);
     }
     return c;
}

vector<int> mul(vector<int> a, vector<int> b, bool &status)//mul function
{
     int i;
     vector<int> c;
```

```
    for (i = 0; i < a.size(); i++)
    {
        c.push_back(a[i] * b[i]);
    }
    return c;
}

vector<int> div(vector<int> a, vector<int> b, bool &status)//div function
{
    int i;
    vector<int> c;
    for (i = 0; i < a.size(); i++)
    {
        //If one element in the second vector is 0
        //store the corresponding element in the first vector
        //into the answer vector.
        if (b[i] == 0)
        {
            c.push_back(a[i]);
            status = 0;
        }
        else
        {
            c.push_back(a[i] / b[i]);
        }
    }
    return c;
}

vector<int> mod(vector<int> a, vector<int> b, bool &status)//mod function
{
    int i;
    vector<int> c;
    for (i = 0; i < a.size(); i++)
    {
        //If one element in the second vector is 0
        //store the corresponding element in the first vector
        //into the answer vector.
        if (b[i] == 0)
        {
            c.push_back(a[i]);
            status = 0;
        }
        else
        {
            c.push_back(a[i] % b[i]);
        }
    }
    return c;
}

void display(vector<int> a, vector<int> b, vector<int> c, char t)//display
function
{
    if (t == '+')
    {
        cout << "\ni: Elementwise Sum" << endl;
        for (int i = 0; i < c.size(); i++)
        {
            cout << i << ": " << a[i] << " + " << b[i] << " = " << c[i] <<
endl;
        }
    }

    else if (t == '-')
    {
        cout << "\ni: Elementwise Difference" << endl;
        for (int i = 0; i < c.size(); i++)
        {
            cout << i << ": " << a[i] << " - " << b[i] << " = " << c[i] <<
endl;
        }
    }

    else if (t == '*')
    {
          cout << "\ni: Elementwise Product" << endl;
          for (int i = 0; i < c.size(); i++)
          {
               cout << i << ": " << a[i] << " * " << b[i] << " = " << c[i] <<
endl;
          }
     }
     else if (t == '/')
     {
          cout << "\ni: Elementwise Quotient" << endl;
          for (int i = 0; i < c.size(); i++)
          {
               cout << i << ": " << a[i] << " / " << b[i] << " = " << c[i] <<
endl;
          }
     }
     else if (t == '%')
     {
          cout << "\ni: Elementwise Mod Operation" << endl;
          for (int i = 0; i < c.size(); i++)
          {
               cout << i << ": " << a[i] << " % " << b[i] << " = " << c[i] <<
endl;
          }
     }
}
int main(int argc, char* argv[])
{
     vector<int> a;
     vector<int> b;
     vector<int> c;
     int n = (argc - 1) / 2;
     bool status = 1;

     for (int j = 1; j <= n; j++)
     {
          a.push_back(stoi(argv[j]));
     }//store the first vector
```

```
    for (int j = n + 2; j < 2 * n + 2; j++)
    {
        b.push_back(stoi(argv[j]));
    }//store the second vector
     //distinguish the symbol
    if (argv[n + 1][0] == '+')
    {
        c = add(a, b, status);
        display(a, b, c, '+');
    }
    else if (argv[n + 1][0] == '-')
    {
        c = sub(a, b, status);
        display(a, b, c, '-');
    }
    else if (argv[n + 1][0] == '*')
    {
        c = mul(a, b, status);
        display(a, b, c, '*');
    }
    else if (argv[n + 1][0] == '/')
    {
        c = div(a, b, status);
        if (status)//bool value pase by reference to see if 0 occur or not
        {
            display(a, b, c, '/');
        }
        else
        {
            cout << "divison by zero occurred for at least one element." <<
endl;
            display(a, b, c, '/');
        }
    }
    else if (argv[n + 1][0] == '%')
    {
        c = mod(a, b, status);
```

```
            if (status)//bool value pase by reference to see if 0 occur or not
            {
                display(a, b, c, '%');
            }
            else
            {
                cout << "mod with respect to zero for at least one element." <<
endl;
                display(a, b, c, '%');
            }
        }
        else
        {
            cout << "Unknown operator! Returning to operating system..." << endl;
            return (-1);
        }
        return 0;
    }
```

2:

```
/****************************
*Filename:parkingfees
*Date:02.01.2019
*Author:Chuang, Tzu-Yi
*matriculation number:4018095
****************************/
#include<iostream>
#include<vector>
#include<string>
#include<iomanip>
using namespace std;
vector<double> calculateCharges(vector<double> hours)//the function to
calculate charge
{
     double hr,ch;
     vector<double> charge;
     for (int i = 0; i<hours.size(); i++)
     {
          hr = hours[i];

          if (hr <= 3.0)
          {
               ch = 2.0;
          }
          else if (hr>=19)
          {
               ch = 10.0;
          }
          else
          {
               ch = 2 + (hr - 3)*0.5;
          }
          charge.push_back(ch);
     }
     return charge;
}
void displayOverview(vector<double> hours, vector<double> charge)//the function
```

```
to display output
{
     double totalhours = 0, totalcharge = 0;
     cout << fixed << showpoint;

     cout << setw(5) << "Car" << setw(15) << "Hours"
     << setw(15) << "Charge\n";

     for (size_t i = 0; i <hours.size(); i++)
     {
          double totalhours = 0, totalcharge = 0;
          cout << setw(3) << i + 1 << setw(17) << setprecision(1)
         << hours[i] << setw(14) << setprecision(2)<< charge[i] << "\n";

     }



     for (int i = 0; i < hours.size(); i++)
     {
          totalhours = totalhours + hours[i];
          totalcharge = totalcharge + charge[i];
     }
     cout << setw(7) << "TOTAL" << setw(13) << setprecision(1);
     cout << setw(13) << totalhours << setprecision(2) << setw(14)<<
totalcharge<< setprecision(2) << endl;
}
int main(int argc, char* argv[])
{
     vector<double> hours;
     vector<double> charge;
     for (int i = 1; i <argc; i++)
     {
          hours.push_back(stod(argv[i]));
     }
     charge = calculateCharges(hours);
     displayOverview(hours,charge);
     return 0;
}
```

3:

```
/****************************
*Filename:simplestats
*Date:03.01.2019
*Author:Chuang, Tzu-Yi
*matriculation number:4018095
****************************/
#include<iostream>
#include<vector>
#include<string>
using namespace std;
int max(vector<int> number)//the function to find max
{
    int maximum = number[0];//set the first number as max
    for (int k = 1; k < number.size(); k++)
    {
        //If the next number is greater than original max
        //set it as the new max.
        if (maximum <= number[k])
            maximum = number[k];
        else
            maximum = maximum;
    }
    return maximum;
}
int min(vector<int> number)//the function to find min
{
    int minimum = number[0];//set the first number as min
    for (int k = 1; k < number.size(); k++)
    {
        //If the next number is lower than original min
       //set it as the new min.
        if (minimum >= number[k])
            minimum = number[k];
        else
            minimum = minimum;
    }
    return minimum;
```

```
}
int main(int argc, char* argv[])
{
     int i=1;
     int num;
     bool t=1;
     int maximum;
     int minimum;
     vector<int> number;
     for (int i = 1; i < argc; i++)
     {
          num = stoi(argv[i]);
          if (num >= 10 && num <= 100)//check if the number is between 10 and
100
          {
               for (int k = i; k > 1; k--)
               {
                    //check all elements before each element
                    //if the same element appear, bool is set to 0
                    //and it will not be saved into the vector.
                    if (num == stoi(argv[k - 1]))
                    {
                         t = 0;
                         break;
                    }
                    else
                         t = 1;
               }
               if (t)
                    number.push_back(num);
          }
     }
     maximum = max(number);
     minimum = min(number);
     for (int j = 0; j <number.size(); j++)
     {
          cout << number[j] << endl;
     }
```

```
    cout << "minimum: " << minimum << endl;
    cout << "maximum: " << maximum << endl;
    return 0;
}
```

4:

```
/****************************
*Filename:numberguess
*Date:05.01.2019
*Author:Chuang, Tzu-Yi
*matriculation number:4018095
****************************/
#include <iostream>
#include <stdio.h>
#include <iomanip>
#include <cstdio>
#include <fstream>
#include <string>
//using namespace std;

std::ifstream cin("input4.txt");
std::ofstream cout("myoutput4.txt");

const int UpperLimitNumber = 100;
const int LowerLimitNumber = 1;
const int UpperLimitTries = 7;
const int LowerLimitTries = 3;
using std::endl;
bool checkNumber(int number)//checkNumber function
{
     if (number >= LowerLimitNumber && number <= UpperLimitNumber)
     {
          return true;
     }
     else
     {
          return false;
     }
}
bool checkTries(int tries)//checkTries function
{
     if (tries >= LowerLimitTries && tries <= UpperLimitTries)
     {
          return true;
     }
     else
     {
          return false;
     }
}
bool setupGame(int &numberRef, int &triesRef)//setupGame fuction
{
     int number = numberRef;
     int tries = triesRef;
     //Loop for number
     do
     {
          cout << "ENTER A NUMBER BETWEEN 1 and 100" << endl;
          cin.clear();
          cin >> number;
          cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
          cout << "You entered: " << number << endl;
          numberRef = number;
          if (cin.eof())
          {
              cout << "You do not want to set a number, ";
              cout << "so you stopped the program" << endl;
              return false;
              break;
          }
     } while (checkNumber(number) == false);
     //Loop for tries
     do
     {
          cout << "ENTER TRIES BETWEEN 3 and 7" << endl;
          cin.clear();
          cin >> tries;
          cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
          cout << "You entered: " << tries << endl;
          triesRef = tries;
          if (cin.eof())
        {
            cout << "You do not want to set tries, ";
            cout << "so you stopped the program" << endl;
            return false;
            break;
        }
    } while (checkTries(tries) == false);

    return true;
}
bool guessNumber(int guessnumber, int correctnumber, int&
triesRef)//guessNumber function
{
    if (triesRef < 1)
        return false;
    if (guessnumber == correctnumber)//guess the correct number
    {
        cout << "With the number " << guessnumber;
        cout << " you guessed the right number and you still had ";
        cout << triesRef << " tries remaining" << endl;
        return true;
    }
    //guess the wrong number
    else if (guessnumber > correctnumber)
    {
        triesRef = triesRef - 1;
        cout << "The number you guessed was too big" << endl;
    }
    else
    {
        triesRef = triesRef - 1;
        cout << "The number you guessed was too small" << endl;
    }
    return false;
}

int makeAGuess(int correctnumber, int& triesRef)//makeAGuess function
{
	int guessnum = 0;
	cout << "Guess the number" << endl;//input the user's guess
	cin.clear();
	cin >> guessnum;
	cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
	if (cin.eof())
	{
		return -1;
	}
	return guessNumber(guessnum, correctnumber, triesRef);
}

int playGame()//playGame function
{
	//In this function the real game start
	int number = 0;
	int tries = 0;
	int guess;
	if (setupGame(number, tries) == false)
	{
		cout << "You do not want to play the game" << endl;
		return -1;
	}
	cout << "The guess number is: " << number;
	cout << " You get " << tries << " tries to find it out" << endl;
	while (makeAGuess(number, tries) != 1)
	{
		cout << "You have " << tries << " tries remaining" << endl;
		if (tries == 0)
		{
			cout << "You ran out of tries" << endl;
			return 0;
		}
		int cont = makeAGuess(number, tries);
		if (cont == -1)
		{
			cout << "You do not want to play the game" << endl;
			return -1;
```

```
            }
        }
        return 1;
}
int main()
{
        int number = 0, tries = 0, guessnum = 0;
        int &numberRef = number;
        int  &triesRef = tries;
        cin >> number;
        cout << "Testing checkNumber(): " << number << " as number" << endl;
        cout << "checkNumber() returned: " << checkNumber(number) << endl;
        cin >> tries;
        cout << "Testing checkTries(): " << tries << " as tries" << endl;
        cout << "checkTries() returned: " << checkTries(tries) << endl;
        cout << "Testing the function setupGame(): " << endl;
        setupGame(numberRef, triesRef);
        cout << "The number to be guessed is: " << numberRef << endl;
        cout << "The player gets " << triesRef << " tries" << endl;
        cout << "Enter your guess: " << endl;
        cin >> guessnum;
        cout << "Testing guessNumber() with: " << guessnum << " as guess ";
        cout << "and: " << numberRef << " as correct number and: ";
        cout << triesRef << " tries" << endl;
        guessNumber(guessnum, numberRef, triesRef);
        cout << "Testing playGame(): " << endl;
        playGame();
        return 0;
}
```