Hochschule Würzburg-Schweinfurt	Prof. Norbert Strobel Florian Spieß	Winter Semester 2018/19	F H W-S

Computing I

C++ Skills Assessment WS 2018-19

(Prerequisite for Final Exam of Computing II)

Problem Statement

Below you find five programming tasks on successive pages. Four of them have to be successfully completed to pass the skills assessment. Deadline is Monday, January 14th, 2019, 24:00.

We expect you to submit a zip file comprising the following results:

1. A report (at least ten pages) structured as follows:
   a. A description of each of the selected programming tasks. You can copy the problem statements listed below.
   b. Screen shots of the output of each of your programs.
   c. A short review of your code. Tell us what you think about your work.
   d. Listings of your C++ code for the selected programming tasks. Please put them into an appendix. This will keep your report more readable.
2. Four files with C++ source code for the four tasks solved. These files carry the extension "cpp". Each source code file must completely implement the solution of the corresponding task. These files must be named as highlighted in green in the subheadings below, e.g., simplecalc.cpp. The inclusion of additional files and libraries besides the C++ standard library is not allowed.
3. Four files executable under Windows. These files carry the extension "exe.", and they must have the names highlighted in blue in the subheadings below, e.g., simplecalc.exe.

You have to complete this assignment on your own. Each of your C++ files submitted has to compile in Visual Studio as available on the school's lab computers in room 1.1.53. Each of your executables must run on these Windows computers as well. We will provide detailed upload instructions during the first week of lectures.

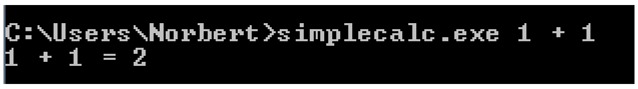
To enable you to check the formatting of your program output beforehand, we provide you with downloadable solutions for predefined program entries. We also provide partially pre-implemented output functions. This should enable you to make your own comparisons and determine whether your programs are correct and the formatting complies with the specifications. Of course, the inputs with which we ultimately check your programs will differ from those that we make available to you initially via download. More details about this procedure can be found below as well.

Programming Tasks

1.) Command Line Calculator

a. Command Line Calculator for individual integer Numbers (*simplecalc.cpp, simplecalc.exe*)

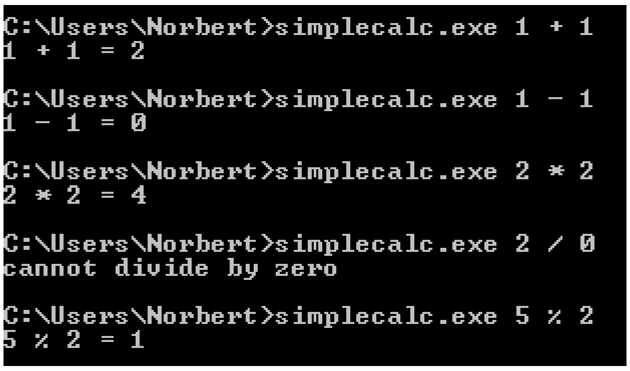
This is about creating a command line calculator. A first number followed by an operator followed by a second number is passed via command line as shown below.



```
C:\Users\Norbert>simplecalc.exe 1 + 1
1 + 1 = 2
```

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`. Please implement a complete solution of the task. Parts of it are already given below.

Your program should work as shown below:



```
C:\Users\Norbert>simplecalc.exe 1 + 1
1 + 1 = 2

C:\Users\Norbert>simplecalc.exe 1 - 1
1 - 1 = 0

C:\Users\Norbert>simplecalc.exe 2 * 2
2 * 2 = 4

C:\Users\Norbert>simplecalc.exe 2 / 0
cannot divide by zero

C:\Users\Norbert>simplecalc.exe 5 % 2
5 % 2 = 1
```

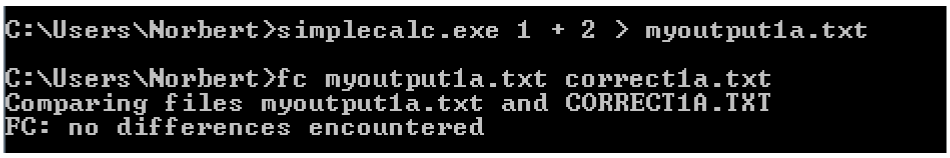
Please note that there need to be spaces between the entries on the command line for things to work well.

C++ Code Snippets

```
//your code here (also abbreviated below as //yours)

//function prototypes
int add(//your code here);
int sub(//your code here);
int mul(//your code here);
int div(//your code here);
int mod(//your code here);

int main(int argc, char* argv[]){

   //your code here (short: //yours)

   //your code here
      cout << //yours << " + " << //yours << " = " << add(//yours)  << endl;
   //your code here
      cout << //yours << " - " << //yours << " = " << sub(//yours)  << endl;
   //your code here
      cout << //yours << " * " << //yours << " = " << mul(//yours)  << endl;
   //your code here
      //yours = div(//yours);
      //your code here
      cout << //yours << " / " << //yours << " = " << //yours << endl;
      //your code here
      cout << "cannot divide by zero " << endl;
   //your code here
      //yours = mod(//yours);
      //your code here
      cout << //yours << " % " << //yours << " = " << //yours << endl;
      //your code here
      cout << "cannot calculate modulo with respect to 0 " << endl;

   //your code here

   return 0;
}
```

Please test that your output is formatted properly by performing the following steps:

1. Run "*simplecalc.exe 1 + 2 > myoutput1a.txt*" on the command line. To this end, enter "*simplecalc.exe 1 + 2 > myoutput1a.txt*" on the command line without "…". Note ">" redirects program outputs to file, here myoutput1a.txt. This is a regular text file that can be opened, e.g., using Notepad.
2. Run "*fc myoutput1a.txt correct1a.txt*" on the command line. "*fc*" is a file comparison program. To do the comparison, download "*correct1a.txt*". If the contents of the two files are identical, then you will get a result similar to what is shown below.

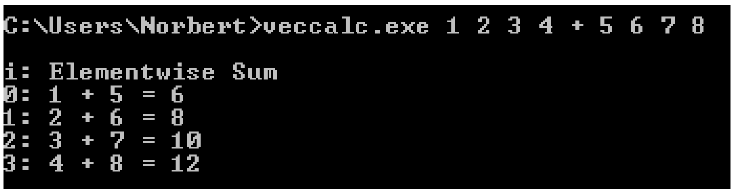


```
C:\Users\Norbert>simplecalc.exe 1 + 2 > myoutput1a.txt

C:\Users\Norbert>fc myoutput1a.txt correct1a.txt
Comparing files myoutput1a.txt and CORRECT1A.TXT
FC: no differences encountered
```

Please note that there need to be spaces between the entries on the command line for things to work well.

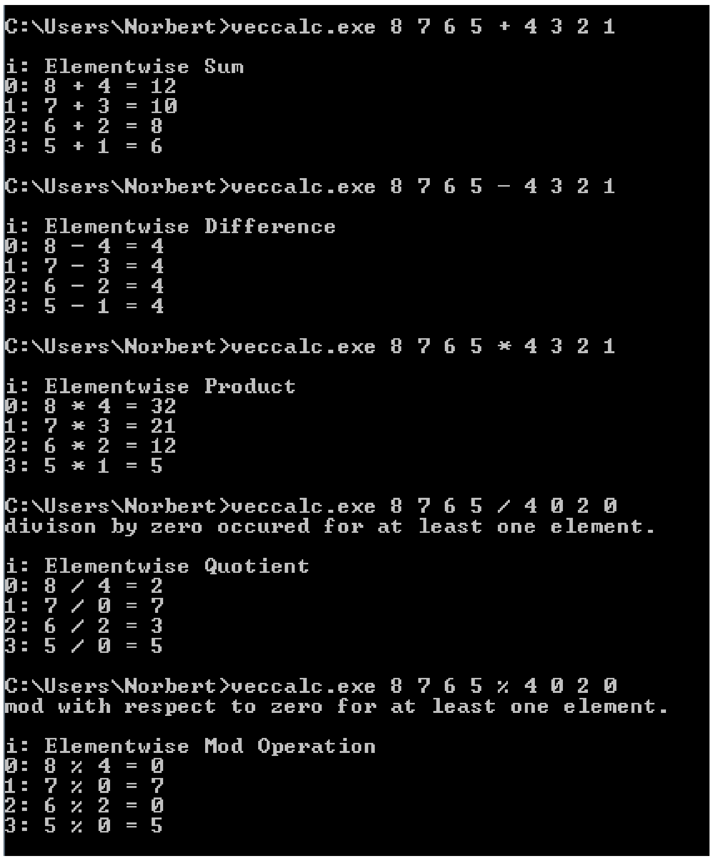
b. Command Line Calculator for Fields of integer Numbers (*veccalc.cpp, veccalc.exe*)

Now write 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 as shown below, e.g., for "+".



```
C:\Users\Norbert>veccalc.exe 1 2 3 4 + 5 6 7 8

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

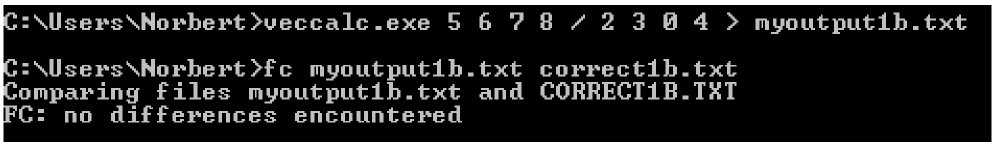
Analogous to *simplecalc*, the operations addition, subtraction, multiplication, division, and modulo are to be implemented. Again, the functions should be called `add(…)`, `sub(…)`, `mul(…)`, `div(…)`, and `mod(…)`, but this time they should operate on fields of type `vector<int>`. Here and below, the 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. Implement a suitable `display(…)` function. Follow the partially implemented function below. Please implement a complete solution of this task. Parts are already given in part. Your output should follow the following screen shot



```
C:\Users\Norbert>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\Norbert>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\Norbert>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\Norbert>veccalc.exe 8 7 6 5 / 4 0 2 0
divison by zero occured 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\Norbert>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
```

Please test that your output is formatted properly by performing the following steps:

1. Run "*veccalc.exe 5 6 7 8 / 2 3 0 4 > myoutput1b.txt*" on the command line.
2. Run "*fc myoutput1b.txt correct1b.txt“* on the command line as well after downloading the file "*correct1b.txt*". If the two files are identical, you receive an output similar to what is shown below.



```
C:\Users\Norbert>veccalc.exe 5 6 7 8 / 2 3 0 4 > myoutput1b.txt

C:\Users\Norbert>fc myoutput1b.txt correct1b.txt
Comparing files myoutput1b.txt and CORRECT1B.TXT
FC: no differences encountered
```

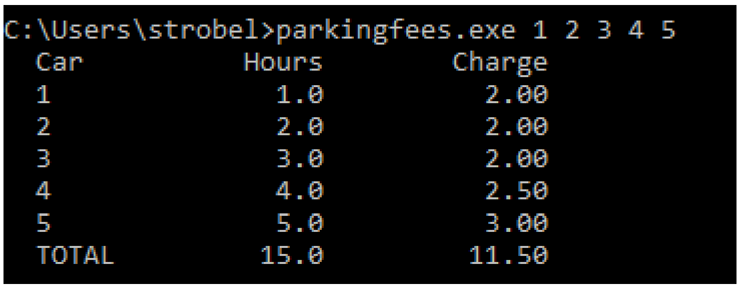
C++ Code Snippets

```
//your code here (also abbreviated below as: //yours)

//function prototypes
vector<int> add(//yours);
vector<int> sub(//yours
vector<int> mul(//yours);
vector<int> div(//yours);
vector<int> mod(//yours
void display(//yours);

int main(int argc, char* argv[]){

   //your code here
   //Note: a, b, c are of type vector<int>
   //you need to figure out their length
   //the variable status is of type bool

   if (//yours) {
      c = add(a,b,status);
      display(a, b, c, "+");
   } else if (//yours) {
      c = sub(a,b,status);
      display(a, b, c, "-");
   } else if (//yours) {
      c = mul(a,b,status);
      display(a, b, c, "*");
   } else if (//yours) {
      c = div(a,b,status);
      if (status)
         display(a, b, c, "/");
      else{
         cout << "divison by zero occurred for at least one element." << endl;
         display(a,b,c,"/");
     }
   } else if (//yours) {;
     c = mod(a,b,status);
     if (status)
        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;
}
```

```
void display(//your code here){

   if (//yours){
      cout << "\ni: Elementwise Sum" << endl;
      for (//yours)
         cout << i << ": " << a[i] << " + " << b[i] << " = " << c[i] << endl;
   } else if (//yours){
      cout << "\ni: Elementwise Difference" << endl;
      for (//yours)
         cout << i << ": " << a[i] << " - " << b[i] << " = " << c[i] << endl;
   } else if (//yours){
      cout << "\ni: Elementwise Product" << endl;
      for (//yours)
         cout << i << ": " << a[i] << " * " << b[i] << " = " << c[i] << endl;
   } else if (//yours){
      cout << "\ni: Elementwise Quotient" << endl;
      for (//yours)
         cout << i << ": " << a[i] << " / " << b[i] << " = " << c[i] << endl;
   } else if (//yours){
      cout << "\ni: Elementwise Mod Operation" << endl;
      for (//yours)
         cout << i << ": " << a[i] << " % " << b[i] << " = " << c[i] << endl;
   } else
      cout << "unknown operator " << endl;

   return;
}
```

2.) Parking Fees (*parkingfees.cpp, parkingfees.exe*)

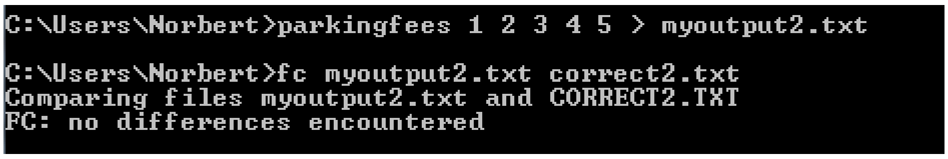
A parking garage charges a EUR 2.00 minimum fee to park for up to three hours. The garage charges an additional EUR 0.50 per hour for each hour or part thereof in excess of three hours. The maximum charge for any given 24-hour period is EUR 10.00. Assume that no car parks for longer than 24 hours at 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. Your 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. Represent the numbers using the data type `double`. Your program should work as displayed below.



```
C:\Users\strobel>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
```

Below you find a partially implemented `main` program. The program should use the function `calculateCharges(…)` to determine the charge for each customer. A function `displayOverview(…)` should generate the table. The function `displayOverview(…)` has been partly implemented already to ensure that the output formatting is consistent. Please test that this is true for your solution as well by performing the following steps:

1. Run "parkingfees.exe 1 2 3 4 5 > myoutput.txt" on the command line (type "parkingfees.exe 1 2 3 4 5 > myoutput.txt" on the command line without the "")
2. Run "fc myoutput.txt correct.txt" on the command line. The file "correct2.txt" can be downloaded. "fc" is the file compare utility under Windows. The desired result is shown below.



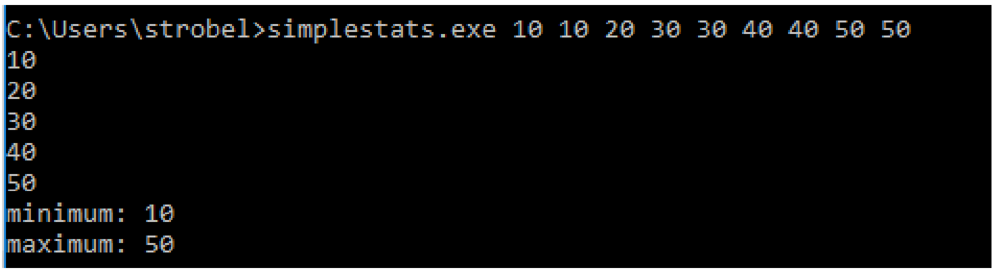
```
C:\Users\Norbert>parkingfees 1 2 3 4 5 > myoutput2.txt

C:\Users\Norbert>fc myoutput2.txt correct2.txt
Comparing files myoutput2.txt and CORRECT2.TXT
FC: no differences encountered
```

C++ Code Snippets

```
//your code here

//function prototypes
//your code calculateCharges(//your code);
//your code displayOverview(//your code);

int main(int argc, char* argv[]) {

   //your code here

   vector<double> hours(//your code here);

   //your code here

   vector<double> charges(//your code here);

   //your code here

   displayOverview(hours, charges);

   return 0;
}

//function definition of displayOverview
displayOverview( //your code here){

   //your code here

   cout << fixed << showpoint;
   // first, display headers
   cout << setw(5) << "Car" << setw(15) << "Hours"
   << setw(15) << "Charge\n";

   for (size_t i=0; i<//your code; i++){
      // display row data for current car
      cout << setw(3) << i+1 << setw(17) << setprecision(1)
      << //your code << setw(14) << setprecision(2)
      << //your code << "\n";
   }

   // display row data for totals
   cout << setw(7) << "TOTAL" << setw(13) << setprecision(1)
   << //your code << setw(14) << setprecision(2)
   << //your code << endl;

   return;
}
```

3.) Detecting and discarding duplicate Entries (*simplestats.cpp, simplestats.exe*)

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 as well (see below).



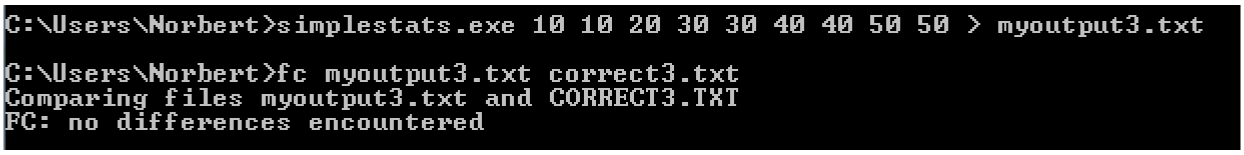
```
C:\Users\strobel>simplestats.exe 10 10 20 30 30 40 40 50 50
10
20
30
40
50
minimum: 10
maximum: 50
```

C++ Code Snippets

```
//your code here

//function prototypes
int min(//your code);
int max(//your code);

int main(int argc, char* argv[]) {

   //your code here


   int minimum  = min(//your code);
   int maximum  = max(//your code);

   cout << "minimum: " << minimum << endl;
   cout << "maximum: " << maximum << endl;

   return 0;
}
```

Please verify that you formatted your output correctly by following the next two steps.

1. Enter "simplestats.exe 10 10 20 20 30 40 50 > myoutput3.txt" on the command line.
2. Enter "fc myoutput3.txt correct3.txt" on the command line. Download the file "correct3.txt" to do the comparison.
3. Desired result (see below)



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

C:\Users\Norbert>fc myoutput3.txt correct3.txt
Comparing files myoutput3.txt and CORRECT3.TXT
FC: no differences encountered
```

4.) Guess-a-Number Game (*numberguess.cpp, numberguess.exe*)

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. Some of them, for example, `makeAGuess(…)`, `setupGame(…)`, `playGame(…)`, have already been implemented to a different degree to ensure a uniform output format.

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.

C++ Code Snippets

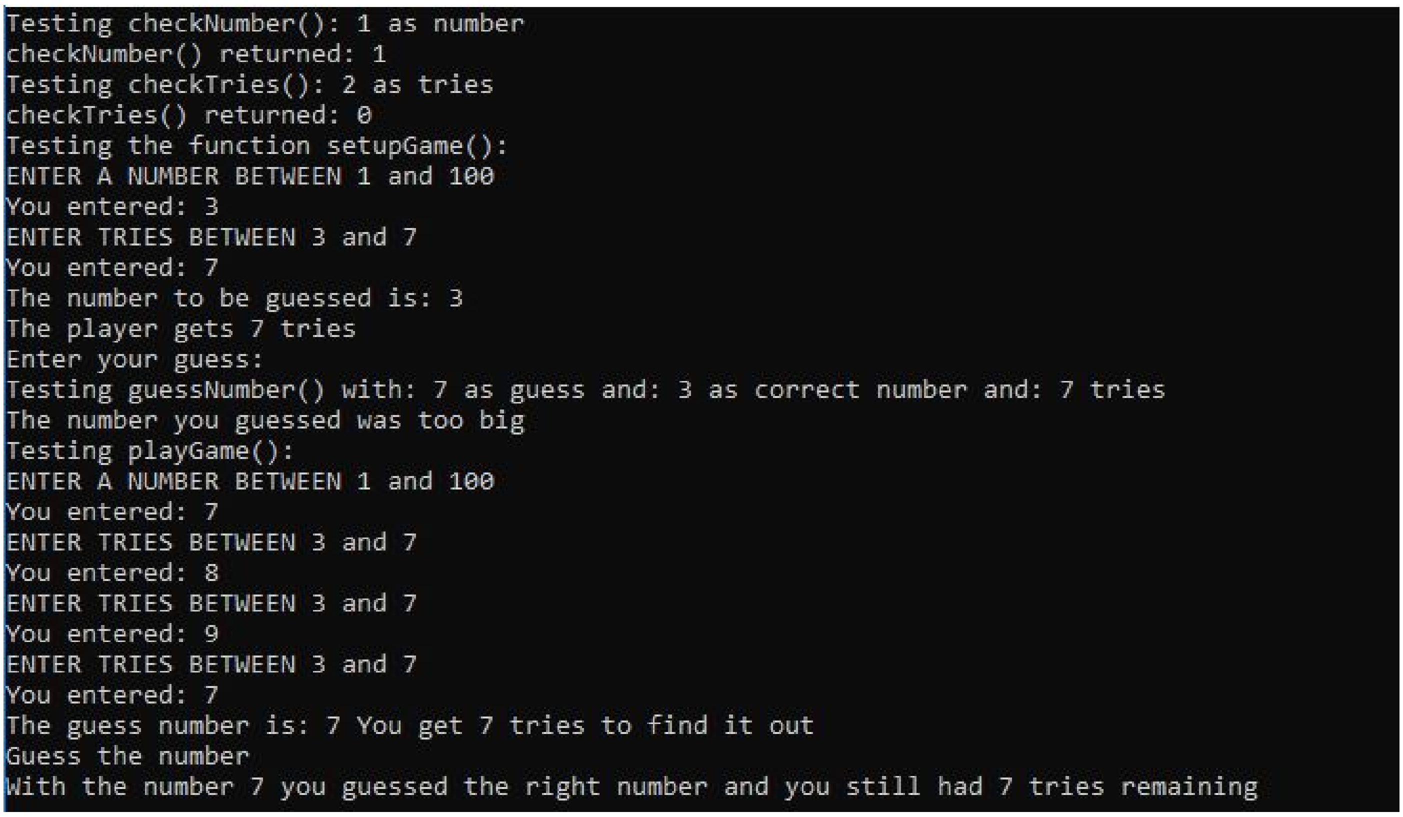
```
#include <iostream>
#include <stdio.h>
#include <iomanip>
#include <cstdio>
#include <fstream>
#include <string>

// define constants
// LowerLimitNumber
// UpperLimitNumber
// UpperLimitTries
// LowerLimitTries
//your code here (below also abbreviated as //yours)

using std::endl;
std::ifstream cin("input4.txt");
std::ofstream cout("myoutput4.txt");
// the three lines just above are to read inputs from file input4.txt
// and to write outputs to file myoutput4.txt instead of using
// cin and cout, respectively.


// function prototypes
bool checkNumber(int number);
bool checkTries(int tries);
bool setupGame(int& numberRef, int& triesRef);
bool guessNumber(int guessnumber, int correctnumber, int& triesRef);
int makeAGuess(int correctnumber, int& triesRef);
int playGame();


int main() {
   int number = 0, tries = 0, guessnum = 0;
   int& numberRef = number, &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;
   // The values associated with numberRef and triesRef will
   // be overwritten inside setupGame(numberRef, triesRef) due to pass
   // by reference. The function setupGame() calls checkNumber()
   // and checkTries()
   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() calls other functions, e.g.,
   // setupGame(), makeAGuess(), etc..
   playGame();

   return 0;
}

bool setupGame(//yours) {
   //your code here (also abbreviated as //yours)
   do {
      cout << "ENTER A NUMBER BETWEEN 1 and 100" << endl;
      cin.clear();
      cin >> //yours
      cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
      //your code here
      cout << "You entered: " << //yours << endl;
      if (//yours) {
         cout << "You do not want to set a number, ";
         cout << "so you stopped the program" << endl;
         //your code here
      }
   } while (//yours);

   //your code here
   do {
     cout << "ENTER TRIES BETWEEN 3 and 7" << endl;
     cin.clear();
     cin >> //yours;
     cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
     //your code here
     cout << "You entered: " << //yours << endl;
     if (//yours) {
        cout << "You do not want to set tries, ";
        cout << "so you stopped the program" << endl;
        //your code here
      }
   } while (//yours);
   //your code here
   return true;
}

bool guessNumber(int guessnumber, int correctnumber, int& triesRef) {
   //your code here
   if (//yours) {
      cout << "With the number " << //yours;
      cout << " you guessed the right number and you still had ";
      cout << //yours << " tries remaining" << endl;
      //yours
   } else if (//yours) {
      cout << "The number you guessed was too big" << endl;
   }
   else {
      cout << "The number you guessed was too small" << endl;
   }
   //your code here
}
```

```
int makeAGuess(//yours) {
   int guessnum = 0;
   cout << "Guess the number" << endl;
   cin.clear();
   cin >> guessnum;
   cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
   if (//yours) {
      return -1;
   }
   return guessNumber(//yours);
}

int playGame() {
//your code here
   if (//yours) {
     cout << "You do not want to play the game" << endl;
     return -1;
   }
   cout << "The guess number is: " << //yours
   cout << " You get " << //yours << " tries to find it out" << endl;
   //your code here
   while (//yours) {
      cout << "You have " << //yours << " tries remaining" << endl;
      if (tries == 0) {
         cout << "You ran out of tries" << endl;
         return 0;
      }
      cont = makeAGuess(//yours);
      if (cont == -1) {
         cout << "You do not want to play the game" << endl;
         return -1;
      }
   }
   return 1;
}
```

You find a screen shot of the game below. You can play the game interactively by commenting out the lines

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

in the C++ source code above.



```
Testing checkNumber(): 1 as number
checkNumber() returned: 1
Testing checkTries(): 2 as tries
checkTries() returned: 0
Testing the function setupGame():
ENTER A NUMBER BETWEEN 1 and 100
You entered: 3
ENTER TRIES BETWEEN 3 and 7
You entered: 7
The number to be guessed is: 3
The player gets 7 tries
Enter your guess:
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
You entered: 7
ENTER TRIES BETWEEN 3 and 7
You entered: 8
ENTER TRIES BETWEEN 3 and 7
You entered: 9
ENTER TRIES BETWEEN 3 and 7
You entered: 7
The guess number is: 7 You get 7 tries to find it out
Guess the number
With the number 7 you guessed the right number and you still had 7 tries remaining
```

5.) Hangman Game (*hangman.cpp, hangman.exe*)

This is now an extension of the last task. It is about guessing a word. In this context you will also encounter something new, namely the handling of C++ strings. First, the player has to enter a word with a minimum of four up to a maximum of 21 characters. The minimum and maximum number of letters should each be defined in a corresponding global variable. Then the number of guessing attempts allowed is queried from the player. Afterwards, the player has to guess the word by entering characters. If the character provided is not part of the word, all letters guessed so far are displayed in the console window and at the same time the number of available guess attempts is reduced by 1. If a correct letter has been typed, the attempt counter will not be decremented. Several functions are required for the game. Some of them have to be implemented completely; others are partially or completely available:

a) `bool checkWord(vector<char> word):`
   This function is used to check whether the `word` to be guessed has the correct length. If the word has up to a maximum of 21 letters (`maxChar`) and at least four letters (`minChar`) , the function returns `true`, otherwise `false`. In this context, introduce the global variables `minChar` (=4) and `maxChar` (=21).

b) `bool checkTries(int tries):`
   This function is used to check the maximum possible number of attempts. If the entered number is between `minTries` (=5) and `maxTries` (=9), the function returns `true`, otherwise `false`. In this context, introduce the two global variables `minTries` and `maxTries`.

c) `bool setupGame(vector<char>& wordRef, int& triesRef):`
   A reference, `wordRef`, is passed to this function that refers to the word to be guessed. It also receives the reference, `triesRef`, which refers to the variable in which the number of guessing attempts is stored. Within this function, the user should be asked to enter the word to guess as well as the number of available attempts. The function should normally be carried out until the inputs meet the conditions set for them. This function first outputs `"Enter your word which should be between " minChar << " and " << maxChar`. Then it reads the associated input via `cin`. Afterwards, the input is confirmed via `"You entered:`

" << testword. Then the input is passed to the checkWord(…) subfunction. Note that you may have to apply the resize(…) function to your vector, e.g., if a wrong word was entered (too short or too long). The function setupGame(…) is to be repeated until a valid input has been made or the user has finished the input via eof (=End of File). This will abort setupGame(…). In this case, setupGame(…)has to return false. In the other case, it displays "Enter your tries which should be between " minTries << " and " << maxTries. A number is read using cin >> testtries. This variable is then displayed to the user via "You entered: " << testtries. Afterwards, the variable testtries is passed to checkTries(…). If the user enters eof (=End of File) (strg+z), the function setupGame(…) ends with the return value false. If, on the other hand, both inputs were received successfully, setupGame(…) returns true.

d) `bool guessAChar(char guessedChar, vector<char> word, vector<char> &alreadyGuessedChars, int& tries, vector<char>& result):`
This function is used to find out whether a letter is part of the word you are looking for. The function should return false if guessedChar is not contained in word. If guessedChar is contained in word, the function returns true. In the positive case, the character is entered in result at the corresponding position. If word contains several identical characters, all of them are to be identified if the corresponding character has been entered once. If a wrong character was entered, the variable tries should be lowered by one. If this value falls below 0 or if a value smaller than 1 is received, the function guessAChar (…) returns false. In addition, the vector alreadyGuessedChars should contain those characters which have already been guessed. It should not happen that a character is stored twice in alreadyGuessedChars.

e) `void showProgress(vector<char> alreadyGuessedChars, vector<char> result):`
After each guessing attempt, the user is to be told which characters he has already guessed and what his previous guessing result looks like. To this end, the vector alreadyGuessedChars is passed to this function. It contains the already guessed characters. These must be displayed in the console window. In addition, a vector result is received. It contains the previous correctly guessed characters at their corresponding positions within the solution. Now use the vector result to output the letters, which have been guessed correctly so far, at their corresponding positions. For places where the letter has not yet been guessed, an underscore (_) must be written. This function has already been fully implemented for you (see below).

f) `int guessWord(vector<char> word, vector<char> alreadyGuessedChars, int tries, vector<char> result):`
In this function, the player is first told how many attempts he still has. To this end, "You have " << tries << " tries remaining" will be displayed. Then you ask the player to enter a character, guess, by displaying "Give a character: ". This character will be confirmed by displaying "You gave " << guess << endl. The character is then passed on to guessAChar(…). After each attempt the participant is shown his current rate of progress via showProgress(…). This process is repeated until all attempts have been used up. The function returns 0 if the participant failed to guess the word. If successful, it returns 1. Don't forget to initialize the variable result within this function. Don't forget that the length (size) of your vector changes when you use the insert(…)function. If guessing is aborted by entering eof (=End of File) using ctrl+z, -1 should be returned.

g) `int playGame():`
Use the setupGame() function to specify the word to guess. Then initialize the variable result with the necessary number of underscores (_). This number corresponds to the number of letters of the word to be identified. The guessWord() function is also used to guess the word. Once a round has been successfully completed, the function returns 1. If there was no success, it returns 0. It returns -1 if the game is stopped in the meantime (eof).

h) As a last step, check whether your game is working as intented by using the supplied input5.txt file. Your program should create a file myoutput5.txt whose content is identical to the content of the downloaded output5.txt file (possibly up to some white space). Since there are numerous text outputs in this task, it cannot be ruled out that small differences may occur that are caused by different spaces and blank lines (white space). Since these differences are not important to us, it is acceptable to suppress white space differences when comparing with "fc". This is done with the call "fc /W". In other words, after calling "fc /W myoutput5.txt correct5.txt", no more differences should be found. Further hints on "fc" can also be found with "help fc".

C++ Code Snippets

```
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <fstream>
#include <string>
#include <vector>

using std::endl;
using std::vector;
using std::string;
std::ifstream cin("input5.txt");
std::ofstream cout("myoutput5.txt");
// the three lines just above are to read inputs from file input5.txt
// and to write outputs to file myoutput5.txt instead of using
// cin and cout, respectively.


//function prototypes
bool checkWord(vector<char> word);
bool checkTries(int tries);
bool setupGame(vector<char>& wordRef, int& triesRef);
bool guessAChar(char guessedChar, vector<char> word, vector<char>& yetGuessedChars,
int& tries, vector<char>& result);
void showProgress(vector<char> yetGuessedChars, vector<char> result);
int guessWord(vector<char> word, vector<char> yetGuessedChars, int tries, vector<char>
result);
bool playGame();
```

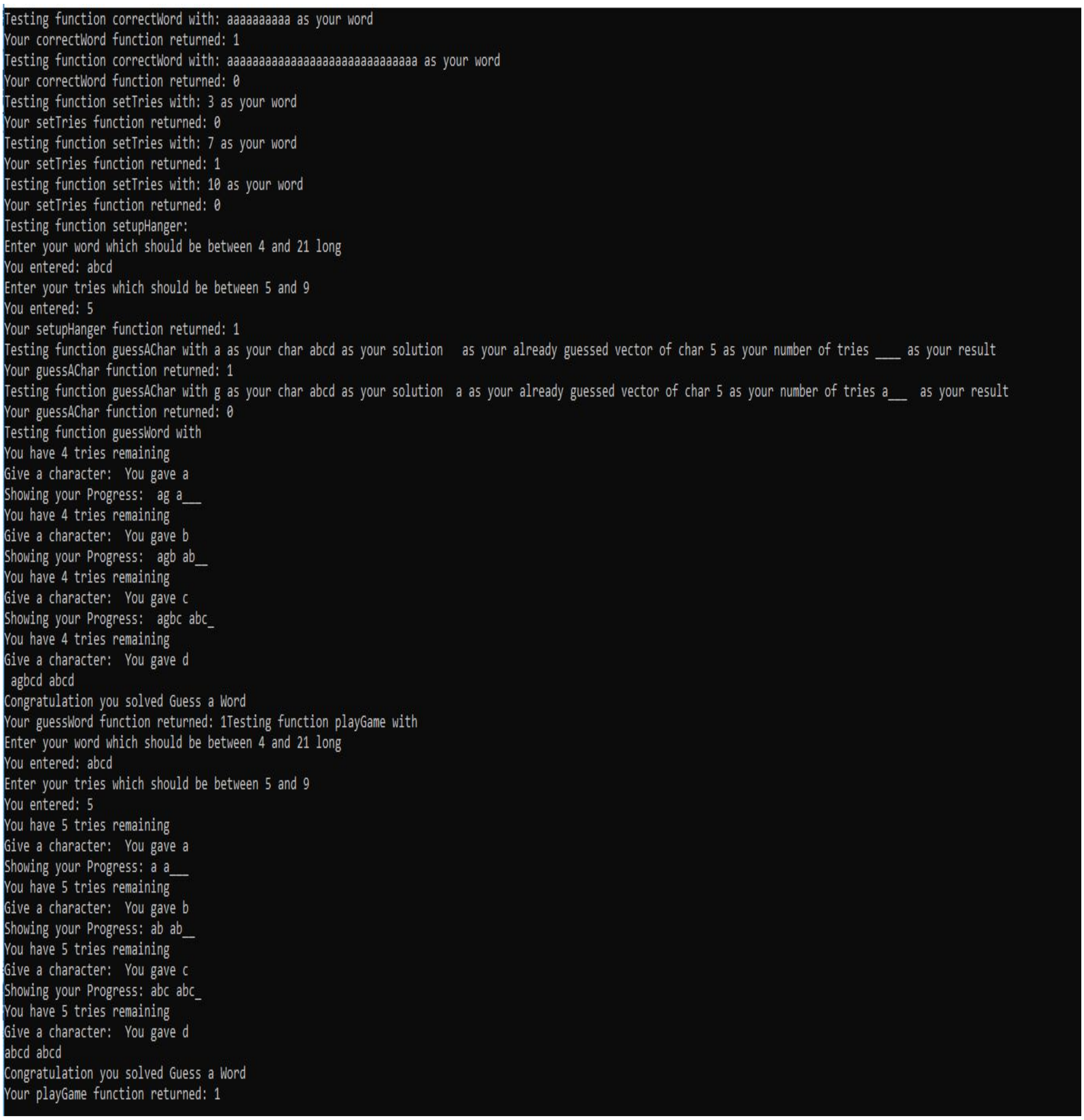
```
int main() {
   vector<char>  testword;
   vector<char>& wordRef = testword;
   int tries;
   int& triesRef = tries;
   testword.insert(testword.end(), 10, 'a');
   std::string str;
   str.resize(testword.size());
   for (int i = 0; i<testword.size(); i++) {
        str[i] = testword[i];//Copy the vector to the string
    }
    cout << "Testing function checkWord() with: ";
    cout << str;
    cout << " as your word" << endl;
    cout << "Your checkWord() function returned: ";
    cout << checkWord(testword) << endl;
    testword.insert(testword.end(), 20, 'a');
    str = string(testword.begin(), testword.end());
    cout << "Testing function checkWord() with: ";
    cout << str;
    cout<< " as your word" << endl;
    cout << "Your checkWord() function returned: ";
    cout << checkWord(testword) << endl;
    tries = 3;
    cout << "Testing function checkTries() with: ";
    cout << tries;
    cout << " as your input" << endl;
    cout << "Your checkTries() function returned: ";
    cout << checkTries(tries) << endl;
    tries = 7;
    cout << "Testing function checkTries() with: ";
    cout << tries;
    cout << " as your input" << endl;
    cout << "Your checkTries() function returned: ";
    cout << checkTries(tries) << endl;
    tries = 10;
    cout << "Testing function checkTries() with: ";
    cout << tries;
    cout << " as your input" << endl;
    cout << "Your checkTries() function returned: ";
    cout << checkTries(tries) << endl;
    cout << "Testing function setupGame()" << endl;
    cout << "Your setupGame() function returned: ";
    cout<< setupGame(wordRef, triesRef) << endl;
    char testchar = 'a';
    vector<char> testguessed;
    testguessed.insert(testguessed.end(), 1, ' ');
    vector<char>& guessRef = testguessed;
    vector<char> testresult;
    testresult.resize(1);
    vector<char>& resultRef = testresult;
    str.resize(testword.size());
    for (int i = 0; i<testword.size(); i++) {
        str[i] = testword[i];//Copy the vector to the string
    }
    std::string string_copy(str);
    string alguess = string(testguessed.begin(), testguessed.end());
    testresult.insert(testresult.begin(), testword.size(), '_');
    string results = string(testresult.begin(), testresult.end());
    cout << "Testing function guessAChar() with: " << testchar;
    cout << " as your char " << string_copy << " as your solution ";
    cout << alguess << " as your already guessed vector of char ";
    cout << tries << " as your number of tries ";
    cout << results << "as your result" << endl;
    cout << "Your guessAChar function returned: ";
    cout << guessAChar(testchar, testword, guessRef, triesRef, resultRef);
    cout << endl;
    testchar = 'g';
    alguess = string(testguessed.begin(), testguessed.end());
    results = string(testresult.begin(), testresult.end());
    cout << "Testing function guessAChar() with: " << testchar;
    cout << " as your char " << string_copy << " as your solution ";
    cout << alguess << " as your already guessed vector of char ";
    cout << tries << " as your number of tries ";
    cout << results << " as your result" << endl;
    cout << "Your guessAChar() function returned: ";
    cout << guessAChar(testchar, testword, guessRef, triesRef, resultRef);
    cout << endl;
    cout << "Testing function guessWord()" << endl;
    cout << "Your guessWord() function returned: ";
    cout << guessWord(testword, testguessed, triesRef, resultRef);
    cout << "Testing function playGame()" << endl;
    cout << "Your playGame() function returned: " << playGame();

    return 0;
}

void showProgress(vector<char> alreadyGuessedChars, vector<char> result){
   for (size_t i = 0; i < alreadyGuessedChars.size(); i++) {
      cout << alreadyGuessedChars[i];
   }
   cout << " ";

   for (size_t i = 0; i < result.size(); i++) {
      cout << result[i];
   }
   cout << endl;
};


bool setupGame(//your code here){

   //your code here (below also abbreviated as //yours)

   do {
      cin.clear();
      cout << "Enter your word which should be between ";
      cout << minChar << " and " << maxChar << " long" << endl;
      cin >> testword;
      cout << "You entered: " << testword << endl;
      data.resize(testword.size());
      std::copy(testword.begin(), testword.end(), data.begin());
      if (cin.eof()) {
         return false;
      }
   } while (checkWord(data) == false);
   wordRef.clear();
   wordRef.insert(wordRef.begin(), data.begin(), data.end());

   do {
      cin.clear();
      cout << "Enter your tries which should be between ";
      cout<< minTries << " and " << maxTries << endl;
      cin >> testtries;
      cout << "You entered: " << testtries << endl;
      if (cin.eof()) {
         return false;
      }
   } while (checkTries(testtries) == false);
   triesRef = testtries;

   return true;
};
```

```
int guessWord(//your code here){


   //your code here
   while (tries >= 1) {
      cout << "You have " << tries << " tries remaining" << endl;
      char guess;
      cin.clear();
      cout << "Give a character: ";
      cin >> guess;
      cout << " You gave " << guess << endl;
      if (cin.eof()) {
         return -1;
      }
      if (guessAChar(guess, word, pgC, ptries, pres) == true) {
         int count = 0;
         for (int i = 0; i<word.size(); i++) {
            if (word[i] == result[i]) {
               count++;
            }
         }
         if (count == word.size()) {
            showProgress(yetGuessedChars, result);
            cout << "Congratulation you solved Guess a Word" << endl;
            return 1;
         }
      }
      cout << "Showing your Progress: ";
      showProgress(yetGuessedChars, result);
   }
   return 0;
};
```

You find a screen shot of the game below. You can play the game interactively by commenting out the lines

```
      std::ifstream cin("input5.txt");
      std::ofstream cout("myoutput5.txt");
```

in the C++ source code above.



```
Testing function correctWord with: aaaaaaaaaa as your word
Your correctWord function returned: 1
Testing function correctWord with: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa as your word
Your correctWord function returned: 0
Testing function setTries with: 3 as your word
Your setTries function returned: 0
Testing function setTries with: 7 as your word
Your setTries function returned: 1
Testing function setTries with: 10 as your word
Your setTries function returned: 0
Testing function setupHanger:
Enter your word which should be between 4 and 21 long
You entered: abcd
Enter your tries which should be between 5 and 9
You entered: 5
Your setupHanger function returned: 1
Testing function guessAChar with a as your char abcd as your solution   as your already guessed vector of char 5 as your number of tries ____ as your result
Your guessAChar function returned: 1
Testing function guessAChar with g as your char abcd as your solution  a as your already guessed vector of char 5 as your number of tries a___  as your result
Your guessAChar function returned: 0
Testing function guessWord with
You have 4 tries remaining
Give a character:  You gave a
Showing your Progress:  ag a___
You have 4 tries remaining
Give a character:  You gave b
Showing your Progress:  agb ab__
You have 4 tries remaining
Give a character:  You gave c
Showing your Progress:  agbc abc_
You have 4 tries remaining
Give a character:  You gave d
 agbcd abcd
Congratulation you solved Guess a Word
Your guessWord function returned: 1Testing function playGame with
Enter your word which should be between 4 and 21 long
You entered: abcd
Enter your tries which should be between 5 and 9
You entered: 5
You have 5 tries remaining
Give a character:  You gave a
Showing your Progress: a a___
You have 5 tries remaining
Give a character:  You gave b
Showing your Progress: ab ab__
You have 5 tries remaining
Give a character:  You gave c
Showing your Progress: abc abc_
You have 5 tries remaining
Give a character:  You gave d
abcd abcd
Congratulation you solved Guess a Word
Your playGame function returned: 1
```