Hangman Game

Name: Abdelrahman Abdallah Shalabi Mohamed Desoki
Neptun code: DOHDZF
Group: 3

1) Developer part

My game consisting of 6 files, named:
- main.cpp
- functionalities.cpp
-functionalities.h
-Intro.cpp
-Draw mistakes.cpp
-words. Text

Let me explain one by one and it is mission, well in **main.cpp** I run the test of my code, and the game by calling the defined base class and derived class,

and it looks like that:

```cpp
#include "Functionalities.h"
#include <iostream>
using namespace std;

int main() {
    //define object for Game, which I will use for accessing all methods
    Game trial;

    trial.data();
    trial.gen_random();
    trial.game_Instruction_storing_data();
    trial.handling_file();
    trial.printwords();
    trial.hidden_words();

    //here  I made another one in case user want to play again
    //first I check then complete with this one
    Intro user;

    while (user.end()){
        user.gen_random();
        user.game_Instruction_storing_data();
        user.handling_file();
        user.printwords();
        user.hidden_words();

    }
    return 0;
}
```

Then in **functionalities.h** I define all the classes I use and methods and attributes: here is the base class which run the game methods in it:

```
class Game {
    //all of this attributes are protected because I inherit the methods
    // which use them in the derived class
protected:
    vector <string> lines;
    int trials=0, mistakes=0, user_length, level, random, cnt=0, size=0;
    char* wordHandling,* dash,* check;
    char user_letters;
    string name ;

public:
    void setvalue(string x) {name= x;}
    string getting (){return name;}
    //here I require two parameters from the user always
    //as it is the main rule of the function
    int  user_input(int start, int end);
    void data();
    void game_Instruction_storing_data();
    void draw_mistakes();
    void printwords();
    void handling_file();
    void hidden_words();
    //here I apply the polymorphism technique
    virtual void gen_random();
};
```

I define all the variables in it as protected to allow them to be inherited by derived class as the user may choose to play the game again, so I need to inherit all of them in that case.

Second thing in **functionalities.h** file is the derived class, which is small one, because most of its methods are already inherited from the base:

```
//Derived class
class Intro :public Game{
//I kept it private as it isn't inherited anymore
private:
    int again;

public:
    void gen_random();
    bool end();

};

#endif /* Functionalities_h */
```

Third part is: functionalities.cpp which almost all the game is handled in it, and also the main algorithm.

Let's take it method by method:

```
10  void Game::data(){
11      //asking the user about his name to call him/her with it during the game.
12      cout<<"please enter your name: ";
13      string h;
14      cin>>h;
15      setvalue(h);
16      cout<<"\n\t\t\tWELCOME "<<getting()<<" TO THE\n\n";
17
18  }
```

Here I am asking the user for his/her name to welcome him/her by their name, then

```
//this function is or printing the logo of the game and instructions
void Game::game_Instruction_storing_data(){
    cout<<"\t\t| *   *    *    *    *   ****    *     *    *    *    * |\n";
    cout<<"\t\t| *   *  *   *  * *  *  *       * * * *  *   *  * *  * |\n";
    cout<<"\t\t| *****  *****  *  * *  * ** *   *  *  *  *****  *  * * |\n";
    cout<<"\t\t| *   *  *   *  *   **  *    *   *     *  *   *  *   ** |\n";
    cout<<"\t\t| *   *  *   *  *    *   ****    *     *  *   *  *    * |\n";
    cout<<"\n\t\t\t\t\t\t\tGAME \n";
    cout<<"made by: DESOKII\n";

    cout<<("\n\n\t You have to guess the letters of the given unknown word,"
           "just letter by letter\n\n"
           "\n\t Hint:  The word may be noun-verb-adjective-pronoun. \n\n"
           "\n\t Every correct letter you guess = 1 point\n\n");
    cout <<endl<<"\t\t\t\t LET'S START"<<endl;
    cout<<("\n Please enter the length of the word you want between 1-14 \n");

    //here I am calling user_input function to check the validity of the user input
    user_length=user_input(1, 14);

    //asking the user to choose the level of dificulty he/she wants
    cout<<"\t Please enter the level of difficulty you want, between 0 and 2 \n"
            "\t 0= Easy=>> you have 9 mistakes only then you will lose.\n"
            "\t 1= Medium=>> you have 6 mistakes only then you will lose.\n"
            "\t 2= Hard=>> you have 4 mistakes only then you will lose.\n";
    //calling it again
    level=user_input(0, 2);
```

Int this method, I am printing the logo of the game ☺, then letting the user know the instructions before playing the game, then asking user about length of word he/she want to guess, after that asking the user about the level of difficulty, to set the game depending on the user choice, after that I assign this choice to the number of trials, to connect it to the mistakes after a while.

```
//in this switch case I assign the value of trials depending on user choice
switch (level)
{
case 0:
    trials=9;
    break;
case 1:
    trials=6;
    break;
case 2:
    trials=4;
    break;
}
```

During the game I ask the user many times to enter a number, so I need to check always when this number is valid or not, so I decided to make this function to save time and not repeat the code every time

```
66 //the main rule of this function is to check the validity of the user input
67 //in it I am applying the exception ahndling technique
68 int Game:: user_input(int start, int end){
69     string length;
70     int n=0;
71     while(1){
72         cin>>length;
73         try {
74             for(int i=0;i<length.size();i++){
75                 //checking if the letter is not digit = character
76                if( !isdigit(length[i])  )
77                   throw "\n \t \t \t \t Invalid Input";
78             }
79             //stoi is a built in function used to transfer between string to int
80             //to complete my check on it as a normal integer
81             n=stoi(length);
82             if(n>end || n<start)
83                 throw "\n \t \t \t \t Invalid number \n";
84              else{
85                  break;
86                 }
87         } catch (const char *s) {
88             cout<<s<<endl;
89             //letting the user know the borders of input I require from him
90             printf("\t >-> Please enter a valid length between (%d-%d)  \n",start,end);
91         }
92     }
93     return n;
94 }
```

And in this function, I use exception handling, to throw an error message if the input is char->invalid input, if it's invalid number then-> invalid number, then I print the limitation for the user to memorize it and check if the input is valid or not. After that I use this function to generate a random number, depending on the time on the laptop:

```
void Game::gen_random(){
    srand((unsigned) time(0));
    random=rand()%3000;
}
```

Because I will need this number to connect it as an iterator to the file of words, and by knowing the size of the file (3001), then I put limits for the generated number to not exceed 3000.

Now I need to handle this file of words (words. Text)

```
void Game::handling_file(){
    string line;
    //file handling and open it to start reading the words from it.
     ifstream fin;
     fin.open("words.txt");

   //know the size of the file and fill into the string vector
     while(getline(fin,line)){
         size++;
         for(int i=0;i<=line.size();i++) {
               if(line[i]>=65 && line[i]<=92)
                   line[i]=line[i]+32;
           }
         //storing the words in the file in the vector named lines
         lines.push_back(line);
     }
    // Close the file
        fin.close();
}
```

So, in this function I open the file then loop through the content of the file then make a nested loop to check if the file has an uppercase letter, to let it be lowercase before saving it in my vector lastly, I close the file.
I need to check the right size of the word and test the game also

```
void Game::printwords(){

//looping to find the word randomly depending on the length and random number.
    //to avoid repetitive words.
    for(int i=random;i<=size;i++){
        if(random==size){
            i=0;
        }
        if(lines[i].length()-1==user_length){
            //this line was just for testing
            cout<<lines[i]<<endl;
            //allocate memory for the size for chosen word
            wordHandling = new char [lines[i].size()+1];
            //then copy it for other usage in the game
            copy(lines[i].begin(), lines[i].end(), wordHandling);
            break;
        }
    }
    //we clear the vector, because if user asked to play again
    //to be able to fill it again
    lines.clear();
    size=0;
}
```

So here I am starting to search in the file from the random number till the end, and to avoid collapse if the random number is = last element in the file I made a condition, to let it be 0 and start from the first, then I copy the right word in a char array called wordHandling, to use it afterwards with dash array. Here we came to the most important function in my game:

```
154 void Game::hidden_words(){
155     int h=0,j=0,p=0,q=0;
156     //define an array of dashes to display in the screen for the user
157     dash=new char [user_length+1];
158     //this array we use it to check if user entered this letter before or not
159     check=new char [mistakes+user_length];
160     for ( h=0; h<user_length; h++)
161     {
162         dash[h]='-';
163     }
164     cout<<"\t\t\t\n";
165     for ( int k=0; k<user_length; k++)
166     {
167         cout<< "{" << dash[k] << "}";
168     }
169     cout<<endl;
170 
171     do{
172         q=cnt;
173         cout<<"\t\t\n please enter a letter between a-z , A-Z \n";
174         //in this loop we check if the input is a valid character or not
175         while(1){
176             cin>>user_letters;
177             if(isalpha(user_letters)){
178                 break;
179             }
180             else{
181                 cout<<"\t\t\n please enter a valid character \n";
182             }
```

First I assign dashes to the dash array, print it, then I loop till the user win or lose, then I loop to check the user entered letter if it's valid or not to not

count mistake on another thing except letters, after that

```
187          user_letters=tolower(user_letters);
188
189          //loop to check if the user entered this letter before or not
190          for(int i=0; i<user_length; i++)
191      {
192          if(check[i] == user_letters)
193          {
194              j=1;
195          }
196      }
197          //letting user knew and asking for another input
198      if(j==1)
199      {
200         cout<<"\n You entered this letter before, please eneter another letter\n\n";
201          j=0;
202          continue;
203      }
204          //assigning the input to check array to know fi its entered before or not
205          check[p]=user_letters;
206          p++;
207
208           //loop to check if the input existing in the word or not
209          for(int u=0; u<user_length; u++)
210          {
211              if(user_letters == wordHandling[u])
212              {
213                  //if it's existed then assign it instead of the dash
214                  dash[u]=user_letters;
215                  cnt++;
```

I use to lower function to lowercase the letter which the user entered then I check if this letter entered before or not, after that I assign it to the array named check, then I loop to check if this letter is existing in the word itself or not, then I print the updated version of dashes with user correct trials, then I check the case of winning when the counter =length of the word, or

the case of lose otherwise.

```
220      for ( h=0; h<user_length; h++)
221      {
222          //print the right guessed letters after user entered them
223          cout<< "{" << dash[h] << "}";
224      }
225      cout<<endl;
226
227          //case of winning, when the counter for correct letters = the length of the word
228          if(cnt ==user_length)
229          {
230              cout<<"\n\t\t\tCongratualations you #WON !! \n";
231              cout<<"your score is: "<<cnt<<endl;
232              break;
233          }
234          //this case to know whether we ned to ask the user again or not
235          if(cnt>q)
236          {
237              continue;
238          }
239          //here handling the mistakes
240          else
241          {
242              mistakes++;
243              cout<<"WRONG TRIAL!"<<endl;
244              //calling the function to draw the simulation for the real game
245              Game::draw_mistakes();
246              cout<<"\t Number of your wrong trials is: "<<mistakes<<endl;
```

Let us turn into **intro.cpp file**, it is really small as it describes the 2 methods of the derived class.

```
7  //here I narrow the scope of choosing random number
8  //to let the probability of having same number to decrease
9  //2500 instead of 3000
10 void Intro::gen_random(){
11     srand((unsigned) time(0));
12     random=rand()%2500;
13 }
14
15 //this function used to make sure if user want to play again or end the game
16 bool Intro::end(){
17     cout<<"\t\t\n\nWould you like to play again? \n";
18     cout<<"\t\t\n Press 1 if you want and 0 if you want to exit the game: ";
19     //check the validity of the input
20     again=user_input(0, 1);
21     if(again!= 1){
22         cout<<"\n\t\t\t\t\t Thanks for playing my Game :)\n";
23         return false;
24     }
25
26     cout<<"\n\t\t\t\t\t Thanks For Playing My Game, I hope you enjoyed it:))\n\n"<<endl;
27     return true;
28 }
```

Well in random function I narrow the scope of this number to decrease the probability of making it similar to the last one, then it is hard for the user to have similar word, then in the end function I ask the user if he/she want to

play again or not, if yes then I will inherit all the methods from the base class in addition to the new random method, but if not then I will end the game. Last file is draw mistakes which is related to functionalities.cpp but I separated it because it is so long, in this file I am defining the method of

drawing the simulation to make game near to the reality

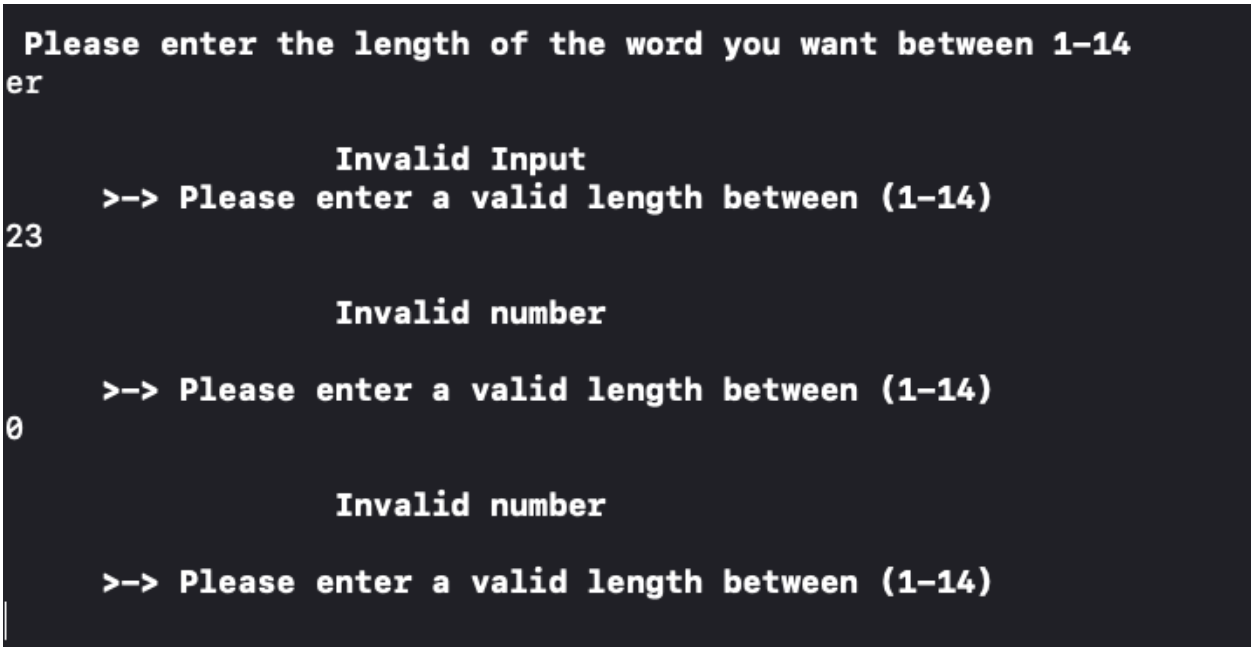
```
void Game::draw_mistakes(){

    //based on easy choice
    if (trials==9){
    switch(mistakes)
    {
    case 1:
        cout << "\n\n";
        cout<<"  |\n";
        cout<<"__|____\n\n";
        break;
    case 2:
        cout<<"  |\n";
        cout<<"  |\n";
        cout<<"  |\n";
        cout<<"  |\n";
        cout<<"__|____\n\n";
        break;
    case 3:
        cout<<"_______\n";
        cout<<"  |\n";
        cout<<"  |\n";
        cout<<"  |\n";
        cout<<"  |\n";
        cout<<"__|____\n\n";
        break;
```

Ending with the man hanged as in real game ☺.

2) Testing part

If we run the project this is the output in the console:

First the user needs to enter the name, then the instructions and logo will be printed, after that user will be required to enter the length of the word



Those inputs are all invalid, so it will keep asking till the input is valid, then if it is valid like this, the game will go on,

```
     >-> Please enter a valid length between (1-14)
5
     Please enter the level of difficulty you want, between 0 and 2
     0= Easy=>> you have 9 mistakes only then you will lose.
     1= Medium=>> you have 6 mistakes only then you will lose.
     2= Hard=>> you have 4 mistakes only then you will lose.
2
```

Okay after that user need to enter the level of difficulty then game must start,

```
{-}{-}{-}{-}{-}

 please enter a letter between a-z , A-Z
7

 please enter a valid character
;

 please enter a valid character
1

{-}{1}{-}{-}{-}

 please enter a letter between a-z , A-Z
```

As the photo describe, it will not count a mistake except it is a valid character, and if the guess is correct then dash will be replaced,

```
y

{-}{1}{-}{-}{-}
WRONG TRIAL!
   |
   |
   |
   |
   |
__|____

     Number of your wrong trials is: 1

 please enter a letter between a-z , A-Z
o

{-}{1}{-}{-}{-}
WRONG TRIAL!
 _______
  |/   |
  |    O
  |
  |
  |
__|____

     Number of your wrong trials is: 2

 please enter a letter between a-z , A-Z
v

{-}{1}{-}{v}{-}

 please enter a letter between a-z , A-Z
```

If the guess is correct, then it will congratulate the user, print the score and ask if he/she want to play again:

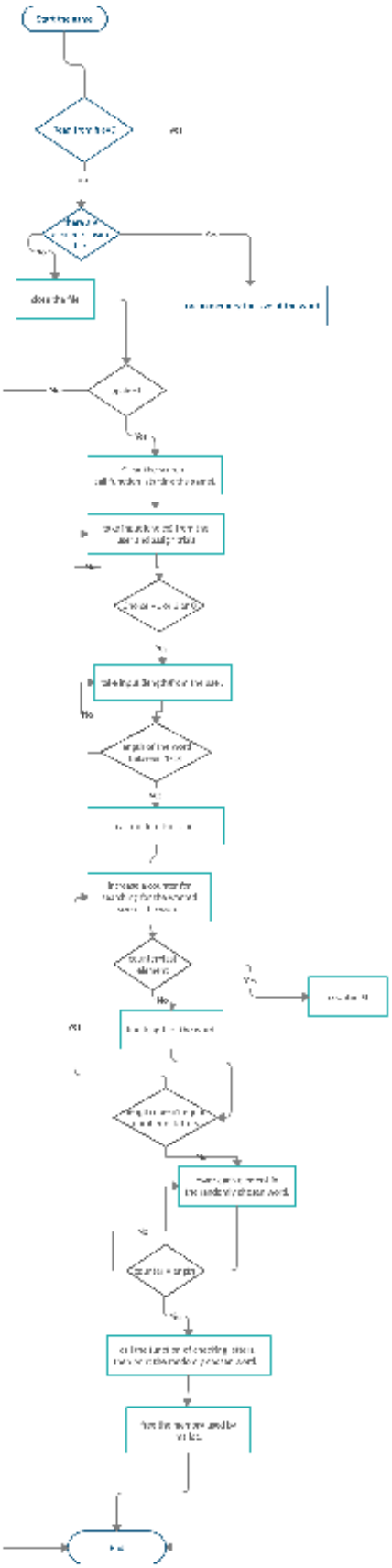
```
{s}{l}{a}{v}{e}

            Congratualations you #WON !!
your score is: 5


Would you like to play again?

 Press 1 if you want and 0 if you want to exit the game:
```

Let us suppose user choose to play again but this time want to lose,

```
n

{-}{-}{-}{-}{-}
WRONG TRIAL!
 _______
  |/   |
  |    O
  |   \|/
  |    |
  |    |
  |   / \
__|____

     Number of your wrong trials is: 4

Unfortunately, you lost all of your: 4 Trials

 your score is: 0
 The word was: least


Would you like to play again?

 Press 1 if you want and 0 if you want to exit the game:
```

Then it will print for the user the score, and also the right word, and ask again the user about playing again.

3)Flowchart of the main idea



It has the general idea of my project, not exactly but quite similar as abbreviation to my project.