# ***Table of Contents***



**Abstract**

Do you love wordle?

Do you love playing Tic-Tac-Toe with that best friend of yours?

Do you love playing rock-paper-scissors when you are tired of doing everything that you must?

Or do you love to test whether you are keeping up with the news around the globe and your favorite celebrities through a refreshing quiz?

We have got you the perfect little program that does all of that. We literally have world famous wordle designed and written all by ourselves with nothing but the little knowledge we have from the C programming.

BK’s Arcade is a game written in C. This program can be installed to a gaming arcade like the ones we see in shopping malls. Users can play the games after paying certain amount and can earn money if they win. It is very interesting to play as a player. It is even more exciting if you look at it as a developer. It has got the authentication system and it creates a unique profile for each player. It allows user to choose between different games.

This project uses almost every aspect of programming from awesome file handling to advanced use of structures and arrays. This program is extremely reusable and readable. The functions, user defined headers and multiple little c files are the highlights of this project. Everything that we learned in class has been used to make a good program.

The program has 6 C files and 5 custom header files, where source code for each game is included in separate individual file. Finally, all the .c files are linked together using different header files (.h).

A data file named “question.txt” is created to store the question of quiz and display it. For wordle, there is a file named “word.txt” that contains the dictionary of words. Authentication also has a data file named “Users.txt” where the login details of the users are stored. As the game takes sensitive data during the signup process, the game is protected with a password where only ten users can log in in a machine.

It's hard to believe that we made these beautiful and secure games only using the basic functions and libraries of C.

**Acknowledgement**

Collaborating with the boys to make something exciting, useful and something that really pushed us to our limit of understating of programming fundamentals was a blissful experience that this project presented us with. Thus, we would like to be grateful to each other at first.

We are thankful to the Department of Computer Engineering for providing us this golden opportunity to work together and create something using the hard gained knowledge of computing over the first semester. We would like to express our gratitude to Ganesh Gautam Sir for teaching us C programming from ground zero and helping us reach the level where we are creating a whole system that runs multiple games.

We are also grateful to all the friends of BCT 2078 and seniors of 2077 for providing us guidance and boosting our confidence when we needed the most.

We must show our sincere respect to known unknown people whose github repositories were source of enlightenment for us.

**Overview**

First of all, this project pushes the beginners to the edge of their limit to think of a problem, find a suitable solution, and then finally write the codes. It doesn't end there; there are uncountable ways in which we, newbie, can mess up. It also teaches the programmers to search for a small bug that is hidden deep inside the code that has the capacity of ruining the whole system. That's why this project was chosen and we are happy that we chose this big messy problem to solve with our newly added coding skills.

1. Wordle: It is a world famous game by New York Times that has millions of daily users. It tests the vocabulary and entertains the players. Wordle can really refresh you and teach you at the same time

2. Quiz: General quiz shuffles the questions on recent exciting news. General knowledge is a must have quality. Our game can inspire people to learn general knowledge.

3. Tic-Tac-Toe: This game here is a mind games as well. Two friends can get real competitive on Tic-Tic-Toe.

4. Rock-paper-scissors: This is very simple yet interesting game that's been around for centuries.

**Applications:**

1. Gaming is one of the biggest industries in our time. So, this can be practically in an arcade shop to attract players to play some simple and refreshing games. Players can get entertained and the shopkeeper can earn his living.

2. Gambling, as evil as it is, is a big thing too. People love betting. So, our program can be used as gambling games as well.

**Objectives**

1. To reach the limit of our programming skills and use all the magical techniques we have in our pockets.

2. To create an exciting and playable game only using the basic tools of C programming.

3. To manage big programs in smaller manageable parts, creating custom headers, dozens of functions and multiple c source files.

**How This Program Works?**

The working process of this program can be explained as:

1. The program begins with a prompt which asks the user to enter the full screen mode and waits for around 2 min and 30 second.

2. After the prompt, a menu appears which provides 3 choices to the user which are:

a. Signup

b. Login

c. Exit

3. If the user chooses Signup option user will be asked to enter some of the his/her personal data and is asked to setup a username and password for login when the user tries to access the game next time.

4. After the signup process is completed or if the user selects login option the user is asked to enter his/her username and password. If the credentials of the user match with the credential recorded in the database, then the user is allowed to proceed to the main menu, otherwise user is asked to re-enter the credentials.

5. If the user selects the exit option the game will simply end with a “Bye Bye :)” prompt on the screen.

6. If the credentials entered by the user matched with the credentials in our database, then the user will be redirected to the main menu where the user can select one of the four games available in the menu.

7. If the user selects the tic tac toe game which is a multiplayer game, then the user will be redirected to the tic-tac-toe screen where the user can select the position to put (x / o) simply by choosing the digit corresponding to the position on the

\_\_|\_\_|\_\_ figure.

\_\_|\_\_|\_\_

| |

8. If the user selects the wordle game the user will be redirected to the wordle screen where the user will have 10 lives and can guess the word for about 10 lives and if the user fails to guess the word the user loses otherwise the user wins.

9. Similarly, if the user selects the quiz option the user will be taken to a sub menu where the user will be able to choose whether the user will be setting up the questions or playing the quiz.

10. If the user chooses to set the question the user will be redirected to the screen where the user will be able to set the questions, else if the user chooses to play the quiz, then user will be able to answer the already setup questions.

11. If the user selects the Rock-Paper-Scissors option in the main menu then the user will be redirected to the Rock-Paper-Scissors screen where the user will be able to select rock or paper or scissors and play against the computer.

12. Finally, after the completion of either of the games, the user will be redirected to the main menu where the user will be able to select the option to exit the game.

**Problem analysis:**

**What is testing?**

It is basically a process using which we verify and validate that an application or software is free of bugs, meets all the technical requirements, abides by all the requirements of development and designing, and meets all the user requirements. Testing ensures that the intended software/ application meets these requirements efficiently and effectively and handles all of the boundary cases and exceptional cases.

**What is debugging?**

It is basically a process using which we fix any bug present in a software or application. In this, we first identify, then analyze, and then remove the errors. Debugging begins after the intended software fails in proper execution. Here, we conclude the problem by solving it and testing the software successfully. This process is considered to be extremely tedious and complex because we need to identify and resolve errors present in every stage of debugging.

**How we debug the code?**

1. Wrote Clean Code.: You do not need to debug code if you don’t write bug.

2. Paid Attention to Error Messages: In almost every development environment, if your code fails, an error message that (to some extent) explains why your code is failing can be seen in the terminal. By seeing these errors, it became easy for us to fix bugs.

3. Googled Things: One of the many amazing things about coding is that the online community is huge. Almost surely there are tons of people already that have faced the same bug you're facing, and that have solved it and explained it so other people don't have to struggle with it, too.

So, If the error message we get wasn't clear to us, or we can't figure out why we're getting it, we googled it.

4. Discussed the problem in the group

5. Narrow Downed Problem and Understood Where the Error is Generate

As our code got bigger, it was hard to analyze every line of code in the search for the bug. So it's a good idea to divide and conquer, starting our search in the places that are most likely to generate the issue.

6. Took a Break and Think about Something Else

Being focused, well rested, and relaxed is key to writing good code and fixing bugs in an effective way. When leaving things alone for a while and thinking about something else, our brain will keep working on the problem on the background, and connecting ideas in a "subconscious" and creative way.

In many occasions it has happened to us that a fresh solution pops into my mind when we see the problem again the next morning. It probably was right there in front of our eyes but because we were tired and stressed we weren't able to see it.

7.Added random Printf lines.: To check on which line the control flow of program reached, we entered random printf lines inside the program where we suspected that error might have occured after reading the error message returned by the compiler. This methodology helped us to detect which line was causing the runtime error and fix it.

At last, after finding the bug we fixed it.

**Findings and Conclusion**

This whole process of writing, testing, debugging and rewriting code over and over for a couple of times was a hell of a ride of all of us. This was one of the biggest

project that we had under our disposal until this point of our life time. Although the whole process was not easy, but with the help of all the teammates and thousands of resources available in the internet, we were able to successfully execute the task. While carrying out this task, we learned how to work as a team. This project not only taught us about the concepts of C programming, but also how to work as a team, how to communicate with the team members, how to divide the tasks among the team members, how to prioritize the tasks based on the difficulty level and many more other things. This arcade is just a demo which includes 4 games only and source code for each game was included in a separate individual file and finally added in the main game through a root file containing the root source code. Thus, this process makes it easier for us to add any newer game. We can simply write code for new game ourselves or we can ask anyone new to write a new code and send us in a file and then we can include the game in our arcade just by calling the master function of the game from our root file contain the root source code. Here the phrase "Root source code" means the source code containing the entry point "Main" function for our code which controls the control flow of whole program. We also encountered different types of error which includes Syntax Error, Run time error and link time errors. And we used different methodologies for debugging the code and making it free from different types of errors which we mention previously.

We figured put an organic idea to work on, found the way to solve the problem, learnt how to collaborate with the fellows. We learn how to divide and conquer using multiple little C source codes and dozens of functions and custom header files. We learnt ways to find what we need from the web.

were able to achieve those objectives and this whole process was fruitful for us!