PROJECT REPORT

On

**“Organ Donation System”**

***Submitted By:***

Yash Kalbhute (CS22030)

***Guided By:***

Mr. Ratnesh K. Choudhary



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**S. B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND RESEARCH, NAGPUR.**

(An Autonomous Institute, Affiliated to RTMNU, Nagpur)

**2023-2024**

**© S.B.J.I.T.M.R Nagpur 2023**

**S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND RESEARCH, NAGPUR**

(An Autonomous Institute, Affiliated to RTMNU, Nagpur)

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

SESSION 2023-2024

#### CERTIFICATE

This is to certify that the Project titled **“Organ Donation System”** is a bonafide work of **Yash Kalbhute** carried out for the partial fulfillment of course work of “Data Structure & Algorithm”, 3rd Semester, Bachelor of Technology in **Computer** Science **& Engineering.**

|  |  |  |
| --- | --- | --- |
| **Mr. Ratnesh K. Choudhary**  Assistant Professor |  | **Dr. Mrudula Nimbarte**  Head of Department |

(Project Guide)

**INDEX**

|  |  |  |
| --- | --- | --- |
| CERTIFICATE | | i |
| INDEX | | ii |
| LIST OF FIGURES | | iii |
|  |  |  |
| CHAPTER 1 | INTRODUCTION | 1 |
| CHAPTER 2 | METHODOLOGY | 3 |
| CHAPTER 3 | TOOLS/PLATFORMS | 5 |
| CHAPTER 4 | DESIGN & IMPLEMENTATION |  |
|  | 4.1 ALGORITHM | 6 |
|  | 4.2 FLOWCHART | 8 |
|  | 4.3 SOURCE CODE | 9 |
| CHAPTER 5 | RESULT & DISCUSSION |  |
|  | 5.1 OUTPUT & DISCUSSION | 16 |
|  | 5.2 APPLICATION | 19 |
| CHAPTER 6 | CONCLUSION | 20 |
|  | REFERENCES | 21 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG. NO.** | **TITLE OF FIGURE** | **PAGE NO.** |
| 4.2.1 | Flow chart-1 | 8 |
| 4.2.2 | Flow chart-2 | 9 |
| 5.1.1 | Welcome screen | 16 |
| 5.1.2 | Getting player name | 16 |
| 5.1.3 | Correct Answer | 17 |
| 5.1.4 | Game 1 | 17 |
| 5.1.5 | Game 2 | 18 |
| 5.1.6 | Score | 18 |

**CHAPTER 1**

**INTRODUCTION**

The Enhanced Organ Donation System is a sophisticated software solution designed to streamline the management of organ donor information and provide a user-friendly interface for both administrators and donors. This system has been comprehensively upgraded to offer enhanced functionality, robust data handling, and improved user experience.

**Key Features:**

**Robust Data Structure**: The system employs a well-defined and flexible data structure that allows for the efficient storage and retrieval of donor information. It accommodates various data types and handles variable-length data, ensuring that all critical donor details can be stored accurately.

**Input Validation and Error Handling:** A pivotal improvement in the system is its stringent input validation and error handling mechanisms. User-provided data, including names, addresses, and other sensitive information, are rigorously validated to ensure data integrity and security. Clear and informative error messages guide users in addressing issues promptly.

**User-Friendly Menu Interface:** The system offers an intuitive menu-driven interface, making it easy for users to add, search, and delete donor records. The improved menu provides clear instructions, ensuring that even non-technical users can navigate the system with confidence.

**Comprehensive Documentation:** The enhanced system includes extensive code comments and documentation to provide a comprehensive understanding of its functionality, data structures, and usage. This documentation assists both developers and users in maximizing the system's potential.

**Security Enhancements:** Security is a top priority in this system. Measures have been implemented to secure file access, protect sensitive user data, and safeguard against potential vulnerabilities. Users can trust that their information is handled with the utmost care.

**Performance Optimization:** The system has been optimized for efficiency, resulting in faster operations and reduced resource consumption. Techniques such as file I/O buffering and strategic data structure usage ensure optimal performance.

**Version Control:** Implementing version control with Git ensures that code modifications are well-managed and reversible. Collaboration among multiple developers is made seamless, fostering teamwork and code integrity.

**Thorough Testing and Validation**: A comprehensive testing plan has been executed to validate the system's functionality. Positive and negative test cases are used to confirm that the system behaves as expected and gracefully handles errors.

**User Training and Support:** Along with deployment, the system includes user training and documentation, empowering users with the knowledge and resources needed to operate the system effectively.

**Continuous Improvement:** The Enhanced Organ Donation System is designed for continuous improvement. It encourages user feedback and feature requests, ensuring that the system remains adaptable and responsive to evolving requirements.

**Data Backup and Recovery:** The system incorporates a robust data backup and recovery plan to protect against data loss, system failures, and unexpected events, providing peace of mind to both administrators and users.

**Security Auditing:** Periodic security audits are conducted to identify and mitigate potential security vulnerabilities or threats to the system, reinforcing its resilience against emerging risks.

In conclusion, the Enhanced Organ Donation System is a cutting-edge solution that not only manages donor information efficiently but also places a strong emphasis on security, user-friendliness, and continuous enhancement. With its key features, it stands as a dependable and adaptable platform to support the critical mission of organ donation.

**CHAPTER 2**

**METHODOLOGY**

**Step 1: Requirement Analysis**

Identify the specific requirements for the organ donation system, including expected input formats, data validation criteria, and error handling requirements.

**Step 2: Data Structure Refinement**

Review and refine the data structure (the struct OrganDonor) to ensure it accommodates all necessary information and supports future scalability. Consider using dynamic memory allocation for strings to handle variable-length data.

**Step 3: Input Validation**

Implement input validation for user-provided data, especially for fields like names and addresses that may contain spaces or special characters. Ensure that user inputs meet defined criteria.

**Step 4: Error Handling Enhancement**

Improve error handling throughout the program. Provide more informative error messages to assist users in understanding and resolving issues.

**Step 5: Documentation**

Add comprehensive comments and documentation throughout the code to explain the program's logic, functions, and data structures. This will make the code more understandable and maintainable.

**Step 6: Menu Improvement**

Enhance the user interface by providing clear instructions and options for each menu choice. Ensure that the menu is user-friendly and provides helpful feedback.

**Step 7: Security Considerations**

Implement security measures to protect user data and the system, such as securing file access, ensuring that user data is not exposed, and protecting against potential vulnerabilities.

**Step 8: Efficiency and Performance**

Optimize the program for efficiency. Consider techniques like buffering file I/O, avoiding redundant file opening and closing, and using appropriate data structures to improve performance.

**Step 9: Version Control**

Implement version control (e.g., Git) to track changes and collaborate with others on the codebase. This ensures that code modifications are well-managed and reversible.

**Step 10: Testing and Validation**

Develop a testing plan to validate the program's functionality, including positive and negative test cases. Ensure that the program behaves as expected and handles errors gracefully.

**Step 11: Deployment and User Training**

Deploy the enhanced program to the intended users and provide training or documentation to help them effectively use the system.

**Step 12: Maintenance and Feedback**

Continuously maintain and update the program to address user feedback and potential issues. Ensure that the system remains reliable and secure.

**Step 13: Future Features**

Consider adding new features or expanding the program's functionality based on user needs and requirements. Continuously gather user feedback to drive further improvements.

**Step 14: Documentation Updates**

Keep the program's documentation up to date as you make changes and improvements. This includes code comments and user guides.

**Step 15: Backup and Data Recovery**

Implement a backup and data recovery plan to protect against data loss, system failures, or other unexpected events.

**Step 16: Security Auditing**

Periodically conduct security audits to identify and address potential security vulnerabilities or threats to the system.

**CHAPTER 3**

**TOOLS/PLATFORMS**

**3.1 SOFTWARE REQUIREMENT**

**a. IDE / FRAMEWORK:**

* Visual Studio
* Dev c++
* turboc++
* ms-word

**b. OPERATING SYSTEM:**

* Windows 11.

**CHAPTER 4**

**DESIGN & IMPLEMENTATION**

**4.1 ALGORITHM**

**Algorithm for Enhanced Organ Donation System:**

**Initialization:**

Load the program.

Initialize the data structure for donor records.

User Interface:

Display a user-friendly menu to the user.

Prompt the user to select an action:

1: Add Donor Information

2: Search Donor Information

3: Delete Donor Information

4: Exit

**Action Selection**:

Based on the user's choice, perform one of the following actions:

**Add Donor Information (Choice 1**):

**Prompt the user for donor details:**

Name

Address

Date of Birth

Gender

Donating Organ

Blood Group

Donor ID

Validate the input data.

If input is valid:

Create a new donor record.

Save the record to the data structure.

Append the record to the data file.

Display a success message.

If input is invalid:

Display an error message.

**Search Donor Information (Choice 2):**

Prompt the user to enter a donor ID to search for.

Search for the donor record with the given ID in the data file.

If the record is found:

Display the donor's information.

If the record is not found:

Display a message indicating that the donor was not found.

**Delete Donor Information (Choice 3):**

Prompt the user to enter a donor ID to delete.

Search for the donor record with the given ID in the data file.

If the record is found:

Remove the record from the data structure.

Create a temporary file.

Copy all records, except the one to be deleted, to the temporary file.

Close and delete the original data file.

Rename the temporary file to the original file name.

Display a success message.

If the record is not found:

Display a message indicating that the donor was not found.

**Exit (Choice 4):**

Close the program.

**Functions:**

In the "Organ Donation System" program, functions are used to modularize the code and perform specific tasks. Here are the functions defined in the program:

**void saveDonorInfo(FILE \*file, struct OrganDonor donor):**

This function is responsible for saving organ donor information to a file.

int searchDonorInfo(FILE \*file, int donorID): This function is used to search for donor information in the file based on the donor ID.

**4.2**

**FLOWCHART**

Initialize variables

gd, gm, count, marks menue()

Call home()

For i = 1 to 13:

clrscr();

ri =I;

Initgraph();

If marks=100

If marks=50

Outtextxy();

Setbkcolor();

getch();

Call game();

Call gamec();

Marks=marks +10;

Print correct in green

Display wrong in yellow

Display score (marks)

Getch() for input

**Fig. 4.2.1**

Check score

if lost

Display you lost the game

Closegraph();

Clrscr();

**Fig.4.2.2**

**4.3**

**SOURCE CODE**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

#include<graphics.h>

#include<dos.h>

# define MS (157.2)\*5

#include<ctype.h>

Void home();

Void gamec();

Void game();

Void main()

{

int count,r,r1,i,n,j;

int gd=DETECT,gm;

home();

count=0;

int marks=0;

for(i=1;i<=13;i++) {

system("cls");

r1=i;

if (count==5||marks==50)

{

game();

}

if(count==10||marks==50)

{

gamec();

}

initgraph(&gd,&gm,"C://turboc3//bgi");

setbkcolor(BROWN);

switch(r1)

{

case 1:

outtextxy(120,170,"1. Which of the following is a Palindrome number?");

outtextxy(150,200,"A.42042 B.101010");

outtextxy(150,230,"C.23232 D.01234");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is C.23232");

getch();

break;

}

case 2:

outtextxy(120,170,"2. Choose the prefix form of (A+B\*C)-D?");

outtextxy(150,200,"A. -A+(BC\*)D B.-+AB\*CD");

outtextxy(150,230,"C.-ABC+\*D D.-+A\*BCD");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is A. -A+(BC\*)D");

getch();

break;

}

case 3:

outtextxy(90,170,"3. Choose the sorting techniques which is not stable ");

outtextxy(230,180,"sorting algorithm??");

outtextxy(150,200,"A.Insertion sort B.Selection sort");

outtextxy(150,230,"C.Bubble Sort D.Merge Sort");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is B. Selection Sort");

getch();

break;

}

case 4:

outtextxy(100,170,"4. What is the worst case time complexity of binary search??");

outtextxy(150,200,"A.O(Nlog N) B.O(N\*N)");

outtextxy(150,230,"C.O(root(N)) D.O(Log N)");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is D.O(Log N)");

getch();

break;

}

case 5:

outtextxy(130,170,"5. Select the worst case time complexity of Shell Sort ");

outtextxy(230,190,"& Merge Sort respectively??");

outtextxy(150,220,"A.O(nlogn), O(N) B.O(logn), O(N\*N)");

outtextxy(150,250,"C.O(n), O(logn) D.O(n2), O(nlogn)");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is D.O(n2), O(nlogn)");

getch();

break;

}

case 6:

outtextxy(90,170,"6. What data structure is used when converting an infix notation");

outtextxy(230,190," to prefix notation?");

outtextxy(150,220,"A.Stack B.Queue");

outtextxy(150,230,"C.B-Trees D.Linked-list");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is A.Stack");

getch();

break;

}

case 7:

outtextxy(80,170,"7. Choose the design technique used in the quick sort algorithm?");

outtextxy(150,200,"A.Dynamic programming B.Backtracking");

outtextxy(150,230,"C.Divide-and-conquer D.Greedy method");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is C.Divide-and-conquer");

getch();

break;

}

case 8:

outtextxy(80,170,"8. Which of the following properties is associated with a queue?");

outtextxy(130,200,"A. First In Last Out B.First In First Out");

outtextxy(130,230,"C.both (a) & (b) D.None of above");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is B.First In First Out");

getch();

break;

}

case 9:

outtextxy(120,150,"9. What are splay trees?");

outtextxy(130,180,"A.self adjusting binary search trees");

outtextxy(130,205,"B.self adjusting binary trees");

outtextxy(130,230,"C.a tree with strings");

outtextxy(130,255,"D. a tree with probability distributions");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(100,300,"Wrong!!! The correct answer is A.self adjusting binary search trees ");

getch();

break;

}

case 10:

outtextxy(80,170,"10. A B-tree of order 4 and of height 3 will have a maximum ");

outtextxy(230,190," of \_\_\_\_\_\_\_ keys.");

outtextxy(150,210,"A.255 B.63");

outtextxy(150,240,"C.127 D.188");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{

setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is A.255");

getch();

break;

}

case 11:

outtextxy(120,170,"11.What is the search complexity in direct addressing?");

outtextxy(150,200,"A.O(n) B.O(logn)");

outtextxy(150,230,"C.O(nlogn) D.O(1)");

if (toupper(getch())=='C')

{

marks= marks+10;

printf("\t\t\t\t\t\t\t\t\n\nMarks: %d ",marks);

setcolor(GREEN);

outtextxy(250,300,"Correct!!!");count++;

getch();

break;

}

else

{ setcolor(YELLOW);

outtextxy(180,300,"Wrong!!! The correct answer is D.O(1)");

getch();

break;

}

}

}

if(count<5||count<50)

{

outtextxy(200,170,"YOU LOOSE THE GAME ! . ! TRY HARDER NEXT TIME");

}

closegraph();

system("cls");

}

**CHAPTER 5**

**RESULT & DISCUSSION**

**5.1 OUTPUT & DISCUSSION**



**Fig 5.1.1** Welcome Screen

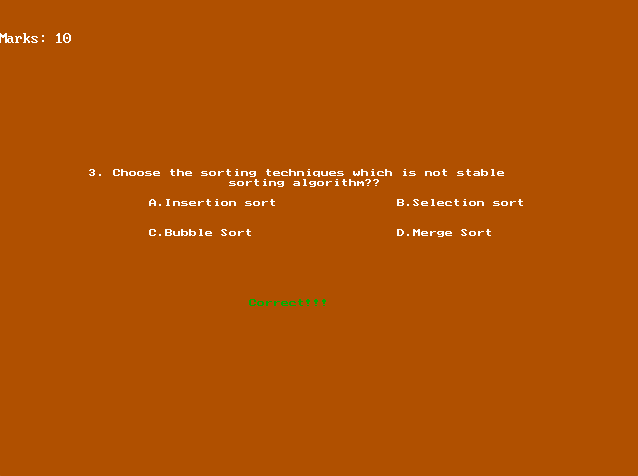
This is the welcome Screen. There are few options like start and quit, this is main home Screen.

You can Enter the game by pressing s or quit it by entering q.



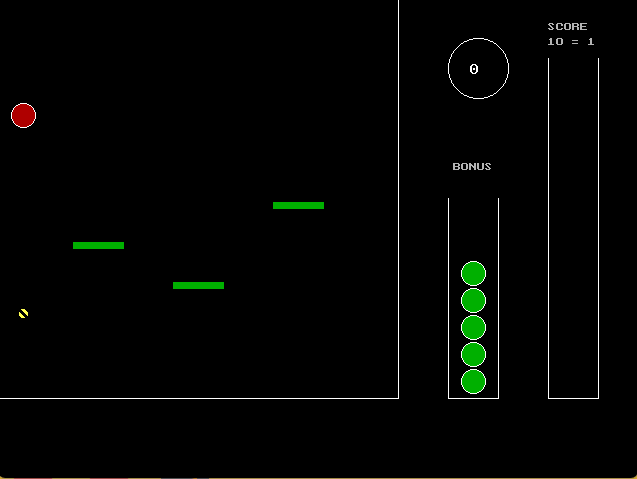
**Fig 5.1.2** Getting player name

This is the second Screen in which you had to enter your name, just to store your score with name for your high score after this screen the quiz will start. We taken input using scanf function and this is make you make trustable and exited about the game.

****

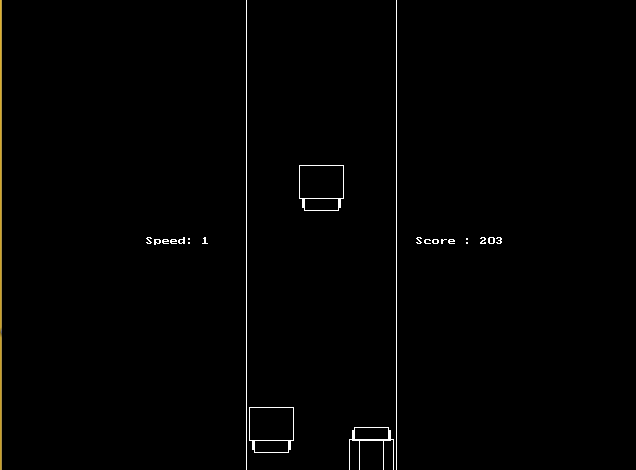
**Fig 5.1.3** Right Answer

Here is the first question with right answer. In this screen you can see question with question number and marks on each question is 10 if you give right answer you will get 10 marks, you can see your answer is correct or not at the bottom of the screen.

****

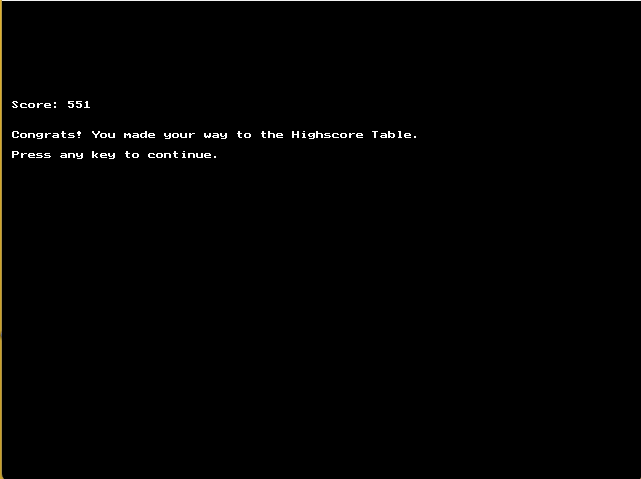
**Fig 5.1.4** Game 1

When you clear the first stage of the game that is if you score 50 marks, you will able to play this game. In this game you had to get the ball with the ball if ball touches the bottom of the screen or upper part of the screen you will loose the chance. For your help there are platforms which lift your ball.

****

**Fig 5.1.5** Game 2

When you clear the Second stage of the game that is if you score 90 marks, you will able to play this game. This is basically a car game but this will give you best game experience. In this game you had to make your car safe from another coming car you get 3 rows and there will increase with increase in score or vice versa, if your car touches to another car you will loose the game.

****

**Fig 5.1.6** score/last screen

This is the end of the game where you can see your score of this game. If you made a high score then there will be the output or display of “Congrats! You made way to high score table”

**5.3 APPLICATION**

* We provide a menu with options to Play Quiz or Quit.
* The user can enter their choice (1 or 2).
* Depending on the choice, the program either starts the game or quits.
* The game and score functions (home and score) remain the same as in the previous code.
* With this application structure, users can interact with the game and play the quiz by selecting option 1. To exit, they can choose option 2. You can compile and run this application to allow users to play the quiz game.
* This program or game is basically design to increase your thinking capacity and for fun

**CHAPTER 6**

**CONCLUSION**

This is simple quiz game where users answer a series of questions to accumulate points. The questions cover a range of topics, including computer science, data structures, and algorithms. The program uses graphics to display questions, options, and user feedback. The user earns points for correct answers, and the game has certain conditions where it transitions to different phases or ends when the user accumulates a specific number of points or answers a set number of questions.

In conclusion, this program is a basic quiz game implemented in C using the Turbo C graphics library. It has potential for improvement and expansion.

In this program there are 2 more programs of games and it takes time to get compiled as it has the more lines of code. But as we can see there is best utilization of time and we can enjoy the games and there will knowledge increase also it will help in MCQ exam practicing the game will provide a refreshment for one and also motivation to play and learn forward. The game is made for fun and for little kids it is way of teaching using of C language and for enjoyment also.

**REFERENCE**

1. <https://www.w3schools.com>
2. <https://www.programiz.com/c-programming>
3. <https://www.geeksforgeeks.org/c-projects/>
4. <https://www.studytonight.com/c-projects/>
5. <https://www.codewithc.com/c-projects-with-source-code/>
6. <https://www.youtube.com/watch?v=yD3MGmGtlWY>
7. <https://www.codewithc.com/quiz-game-mini-project-in-c/>

**Instructions for making project report:**

1. **Chapter Title –** Font size = 14, Font Type = Times New Roman, Alignment = Centered,

Bold.

(Example –**CHAPTER 1**)

1. **Chapter Name –** Front size = 16, Font Type = Times New Roman, Alignment = Centered,

Bold.

(Example –**INTRODUCTION)**

1. **Heading –** Font size = 14, Font Type = Times New Roman, Alignment = Left, Bold.

(Example – **3.1 SOFTWARE REQUIREMENT**)

1. **Sub Heading -** Font size = 12, Font Type = Times New Roman, Alignment = Left, Bold.

(Example – **MODULE, FEATURES**)

1. **Content -** Font size = 12, Font Type = Times New Roman, Alignment = Left.

(Example – Write down proper information about you project)

1. **Figure –** Give proper name to your figure and screenshot.

Front size = 12, Font Type = Times New Roman, Alignment = Centered.

(Example – **Fig. 4.1.1** Flowchart of Tic-Tac-Toe)

1. **Spacing** between all contents =1.5, Alignment= Justify and Text color=Black.