QUIZ USING LINKED LIST

Team Members:
Gurnehmat Kaur Dhindsa
Rashi Maheshwari
Rishitha Korrapati
Darshita C

Reg. No
Reg. No 19BCE0227
Reg. No 19BDS0006
Reg. No 19BCE2275
Reg. No 19BCE2246

Report submitted for the
Final Project Review of

Course Code: CSE2003 – DSA

Professor: Dr. Ilanthenral Kandasamy

Deadline: 5th June 2020

I. Introduction:

Online learning has become an integral part of our lives. This includes taking quizzes, online tutorials, using online reference materials, etc. Online quizzes are really helpful as they can be taken at any time and at any place. In the time of this pandemic we often take quizzes that use web development technologies such as MongoDB to store data and JavaScript libraries to make our experience interactive and enable the storage of our progress and data over the cloud or locally. Data structures and algorithms are an essential part of providing and efficient and smooth experience for the user and for the programmer.

Taking this as our inspiration we decided to make a quiz with technologies that we have learnt over the duration of our semester in this course. This quiz does not compare to web development technologies widely but instead uses technologies we are familiar to and concepts we have learnt in class. We chose linked lists as the basic data structure for this project as it forms the foundations of several projects in the history of programming. The idea of dynamic data structures really fascinated us, so we combined it with the file functionality provided in c++ which allows us to store data permanently, similar to SQL or MongoDB in web development. This project may seem over simplistic due to its use of elementary coding concepts, but it has enabled us to implement technologies we learnt in class in a functional way for real time use. It is a stepping stone in our journeys to become programmers that work on real life problems.

The project displays a menu which enables the user to either play a new quiz game consisting of 10 questions for which the marks are displayed at the end of the quiz. As per the wishes of the user, a detailed report which tells them the correct answers and the answers they gave to all the questions is displayed. The user can also check all the previous logs, or check the logs of a particular person. Another important feature is that this quiz can be taken an umpteen number of times as an infinite menu program is used.

We have used file handling to store the questions, answers and the logs of previous users. A file containing the dataset is read and the questions and their answers are stored as nodes in a linked list. As opposed to a fixed memory data structure such as an array the memory is allocated dynamically. The questions, correct answers, and the entered answers are again stored dynamically in a progress report in the form of a linked list, which the user can choose to display at the end. The logs are appended into a separate file and accessed and updated as per the requirements of the program.

The program created by us uses the object-oriented programming concept of modularity which has enabled us to reuse code by creating modules and running them repeatedly using loops, which has helped us enhance our algorithm. We have learnt a lot about the significance and the implementation of the linked list data structures and the several ways in which it can be done during the course of this project.

II. Literature Review Summary Table

We have hereby reviewed and analysed five research papers related to linked lists, file handling and C++ language which are discussed below-

Authors and Year (Refere nce)	*Title (Study)*	*Concept / Theoretica l model/ Framewor k*	*Methodolo gy used/ Implement ation*	*Dataset details/ Analysis*	*Relevant Finding*	*Limitations / Future Research/ Gaps identified*
Ankil Dalal & Ankur Atri, 2014[1]	*An introductio n to linked list*	*General overview of the history, types, pros& cons and operations of linked lists*	*The implement ation chosen by us was that of pointers to curate linked lists with unidirectio nal links. Nodes were created and traversal and insertion operations were carried out.*	*Linked lists are dynamic data structures, for which the memory is allocated when needed and not previously defined. This memory is freed when needed*	*A singly linear linked list is divided into two parts. The first one contains the data or informatio n while the second contains the address of the next node.*	*Linked lists require a lot of space as the link field, i.e., the field that contains the address of the next node takes up a lot of space*
Karuna & Garima Gupta, 2014[2]	*Dynamic Implement ation Using Linked List*	*Implement ation of linked list operations of insertion at the beginning, insertion at the end, insertion of a node after a node,*	*Structures and pointers have been used to implement various operations on singly linked list, doubly linked list and*	*The time complexity of implementin g a linked list shall never exceed O(n^2) and linear time can be achieved if there is only*	*Singly linked linear lists are extremely useful structure for automatic ally allocating and de-allocating space in a*	*The code and complexity of such algorithms is more than that for an elementary data structure as an array.*

[1] https://journals.pen2print.org/index.php/ijr/article/view/963/910
[2] https://ijermt.org/publication/11/IJERMT%20V-1-5-6.pdf

		deletion and concatenation	*circular linked list.*	*one node or entity.*	*list as per our requirements.*	
Nick Parlante, 1998[3]	*Linked List Basics*	*This document provides a realistic, applied & pointer-intensive approach to understanding linked lists.*	*Extensive explanations, sample code, drawing and exercises using C syntax have been provided in the said article.*	*Detailed illustrations and exegesis has been given about pointers in sections covering linked lists basics- arrays and pointers, list building, coding techniques and examples along with other implementations.*	*The pointer-intensive approach helped in comprehending the working of linked list programs, and finally implementing them on our own in the project.*	*This document provides information about other implementations which can be explored by us in the future, such a doubly linked list, dynamic array, chunk list and head array.*
Devishree Naidu, & Abhishek Prasad[4]	*Implementation of Enhanced Singly Linked List Equipped with doubly linked list operations : An Approach towards Enormous Memory Saving*	*a singly linked list itself to implement operation of doubly linked list as well as singly linked list. This approach makes use of EX-OR Technique to acquire next node and previous node.*	*An EX-OR technique is used to implement a double linked list using a single linked list without increasing the memory requirement.*	*The unidirectional traversal property of the singly linked list is a drawback that can be overcome by using doubly linked lists, but an alternative approach using EX-OR operations can be implemented as*	*We learnt a new method of implementing doubly linked lists through the help of this document.*	*This technique can be used to keep track of processes for their efficient performance in operating systems.*

[3] http://cslibrary.stanford.edu/103/LinkedListBasics.pdf

[4] http://www.ijfcc.org/papers/276-E1045.pdf

				suggested in this paper.		
Bjarne Stroustrup, 1998[5]	*The overview of C++ programming, and language-technical concepts*	*Object oriented programming concepts such as abstraction, polymorphism and encapusulation are talked about in this paper along with the design, evolution and standard library of C++*	*The implementation of the concepts is done in a modern and comprehensible way, thus making it such a useful and foundational language in the world of technology.*	*Important implementational aspects of C++ have been states along with the algorithmic approach that can be taken for large scale programming.*	*Object oriented programming approach, automatic memory allocation, and pointers have been used in our project, which are elaborated upon extensively within the said document.*	*It is complex to use and debug in large high level programs as well as web applications. Other alternatives such a python are being adopted in recent times due to less complexity.*

III. Objective of the project:

The objective of simulating this quiz was to implement the data structures and algorithms' technologies learnt over the last semester, in a practical way and understand the real-life applications and significance of what we do in class. Our aim was to use linked lists in a functional way to create a project that holds relevance to our lives and helps us get hands on experience in using the aforementioned tools.

IV. Innovation component in the project:

The project component of this course has helped us move away from conventional methods of teaching and get hands on experience by simulating a quiz, which is something we come across in our daily lives. The most innovative part of our project has to be the fact that it helps us mimic technologies we use in more simple manner while providing us experience on using the tools we have learnt in this class.

V. Work done and implementation

[5] https://www.researchgate.net/publication/2334185_An_Overview_of_the_C_Programming_Language

Methodology adapted/ Algorithm:

The working of this project consists of the following processes:

1. **Start of the program:** A menu is displayed to give the user an option to play a new game, view all the previous logs, view a particular log or quit the game. Using a switch statement different functions are performed based on the entered choice.

   A structure named **question_storage** containing a question, answer and a self-pointing pointer is created. This acts as the node of a linked list; this will help store the questions and their correct answers. Another structure called **answers_storage** is created as a node for a separate linked list, it will help store questions, the correct answers and the answered entered by the user.

   A file named **questions.txt** was created to contain the dataset. It consists of each questions followed by an answer in the next line.

2. **Playing a new game:**

2.1 The game is played by calling a function named **playGame()** which accepts the name of the user and commences the game.

2.2 A method named **getQuestions()** is called. It accesses a previously created file called **questions.txt** and reads each question and its answer and passes it to the method **QuesInsert()** which in turn inserts it into a linked list of which each node is of the structure **question_storage.**

2.3 The questions are asked using a function **askQuestion()** which accepts the question number and displays a question by accessing the linked list containing all the questions & answers. It takes the answer from the user and checks the correct answer against the entered answer and subsequently changes the score, which is contained in a global variable called **game_score.** The question, its correct answer and the answer entered by the user is stored in another linked list using a function **SaveData().** This process is repeated ten times, i.e, the number of questions.

2.4 The logs are updated using the function **updateLogs()** which appends a file containing all the logs called **logs.txt** and stores the name of the user and their score

2.5 The user is given the choice to display their progress or not. If they say yes, then a function called **traverseStructure()** is called, which traverses the previously created linked list and displays a detailed report.

3. **Displaying all the logs:** This is done using the function **showAllLogs()** which accesses the **logs.txt** file and reads and subsequently prints all the logs, using basic file handling functionality.

4. **Displaying the logs of a particular player:** This is done using a function **showSpecificLog()** which reads the file **logs.txt** and checks where the given name appears and prints all the logs of the said person.

Requirements: A C/C++ compiler such as Dev C++, Turbo C, Code Blocks or Visual Studio Code with an extension can be used to implement this program. The data set is stored on the form of .txt files. We have used Visual Studio Code due to its interactive user face and ability to easily connect to GitHub.

A computer with basic functionality and ability to process the latest technologies can be used, as the data set used is relatively small.

Dataset used:

The dataset has been curated from a C++ quiz on TutorialsPoint.com[6] and GeeksforGeeks.com[7] and is use in the form of a .txt file

Tools used

The programming language used in this project is C++. Its object-oriented approach and its low-level manipulation, which makes the implementation smooth and efficient. We have used the following

The following C++ tools were used during the implementation of this project:

1. **Linked list:** A linked list is a contiguous collection of nodes where each node contains a data field and a reference(link) to the next node in the list which is a self-pointer. A node can be inserted at the beginning, in the middle or at the end of the linked list. Similarly, a node can be deleted from the beginning, from the middle or from the end. Data of each node or a particular node present in the linked list can be displayed by traversal of the linked list from either end. A linked list has been used in this project to help easy removal of nodes, without ruining the flow of the structure.
2. **Queue:** A queue is an abstract data structure which is open for addition and removal at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue)[8]. Queue follows First-In-First-Out methodology (FIFO), i.e., the data item stored first will be accessed/ removed first and a data item will always be added at the last. It is dynamic in nature and the size can be varied according to the discretion of the programmer. Queue can be implemented using linked list, which is what we have employed in this program. We have chosen to use a queue to store the progress of a user as is consists of nodes which contain variables of different data types; besides it can be extended at any point of time due to its dynamic nature.
3. **Files:** Files are used store relevant data at one place in a computer. C++ facilitates file handling, which includes creating, opening, reading, writing, append, and other such functions. A file object of ofstream, ifstream or fsrream class is created, which can be

[6] https://www.tutorialspoint.com/cplusplus/cpp_online_quiz.htm
[7] https://www.geeksforgeeks.org/c-programming-multiple-choice-questions/
[8]https://www.tutorialspoint.com/data_structures_algorithms/dsa_queue.htm#:~:text=Queue%20is%20an%20abstract%20data,first%20will%20be%20accessed%20first.

used to open a previously existing file or creating a new one in several different modes as per the choice of the programmer. We have utilised files in our program to store our data set and retrieve it in every successive use of the program, and another file to store previous and new logs in the same file using append mode.

We have extensively used online resources apart from a few books including ‘Introduction to algorithms- Third Edition’ by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, ‘Data Structures and Algorithms Analysis in C++’ by Mark Allen Weiss, and ‘Computer Science with c++’ by Sumita Arora.

VI. Screenshot and Demo:

Code:

```
#include<iostream>

#include<stdlib.h>

#include<fstream>

#include<string>

#include <cstring>

 using namespace std;

//Global variable to store the score

int game_score=0;


//structure for linked list to contain the questions and answers taken from the file

struct question_storage{

   string question;

   string answer;

   struct question_storage* next;

}*head=NULL, *current, *temp;


//structure for linked list to contain the report of the user
```

```
struct answers_storage{
   string given_question;
   string correct_answer;
   string given_answer;
   struct  answers_storage* next;
}*first=NULL, *ptr1, *ptr2, *head2;

//function to insert Questions into a linked list by reading a file
void QuesInsert(string ques, string ans)
{
   if(head==NULL){
      temp=(struct question_storage *)malloc(sizeof(struct question_storage));
      current=(struct question_storage *)malloc(sizeof(struct question_storage));
      temp->question=ques;
      temp->answer=ans;
      temp->next=NULL;
      head=temp;
      current=head;
   }
   else{
      temp->question=ques;
      temp->answer=ans;
      temp->next=NULL;
      current->next=temp;
      current=temp;
   }
}
```

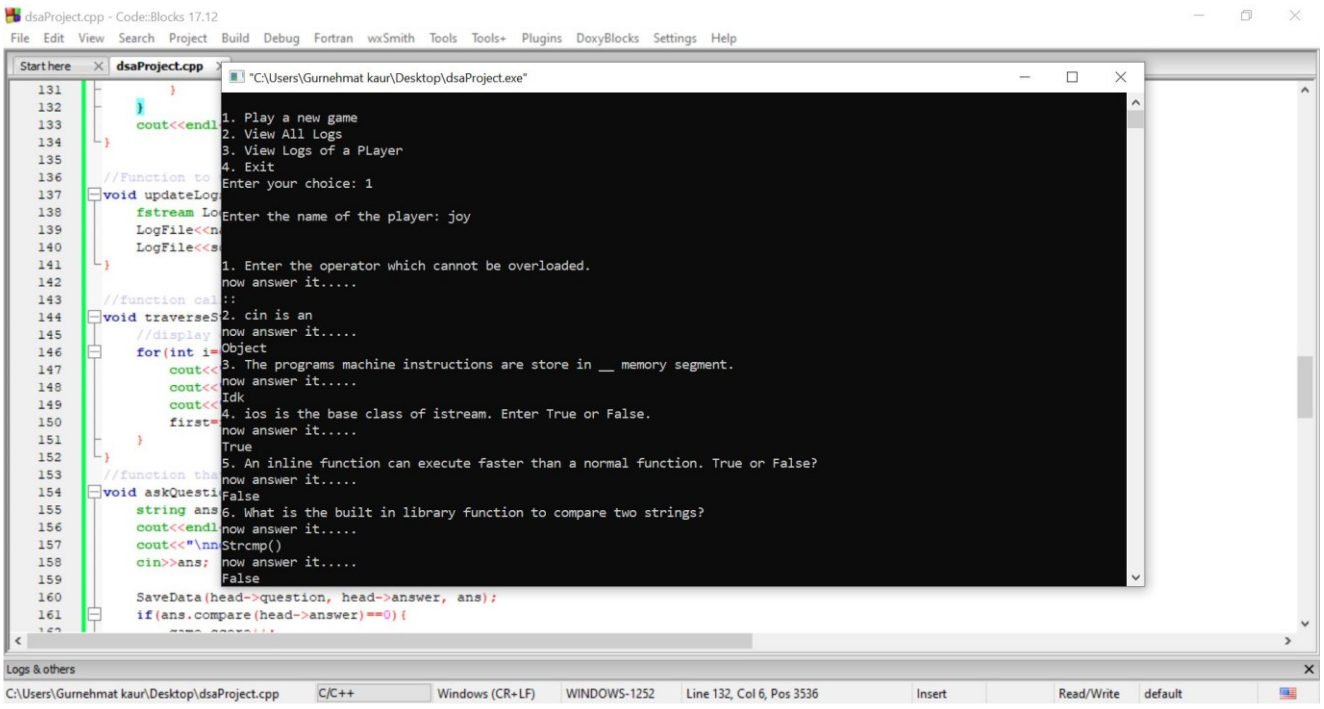
```
//function to store progress report into a linked list
void SaveData(string ques, string cans, string gans){
   if(first==NULL){
      ptr1=(struct answers_storage *)malloc(sizeof(struct answers_storage));
      ptr2=(struct answers_storage *)malloc(sizeof(struct answers_storage));
      ptr1-> given_question=ques;
      ptr1-> correct_answer=cans;
      ptr1-> given_answer=gans;
      ptr1->next=NULL;
      head2=ptr1;
      ptr2=head2;
   }
   else{
      ptr1-> given_question=ques;
      ptr1-> correct_answer=cans;
      ptr1-> given_answer=gans;
      ptr1->next=NULL;
      ptr2->next=ptr1;
      ptr2=ptr1;
   }
}

//Function to read that file and load it onto memory in the form of a linked list
void getQuestions(){
   ifstream file("questions.txt");
   string line;
         cout<<"\n\ntrying to get questions";
```

```
    int i=0;
    string q,a;
    while(getline(file,line))
        {
            if (i==0)
            {
                q=line;
                i++;
            }
            else
            {
                a=line;
                QuesInsert(q,a);
                i=0;
            }
            //insert second line in node and then create a new node for next time and
        }
}

//Function to display logs of all Users
void showAllLogs(){
    std::fstream logFile;
    logFile.open("logs.txt", ios::in);
    std::string logLine;
    int i=0;

    while(getline(logFile, logLine)){
        if(i%2==0){
            cout<<"Name: "<<logLine<<endl;
            i++;
        }
        else{
            cout<<"Score: "<<logLine<<endl<<endl;
        }
    }
}
//Function to display logs of one specific user
void showSpecificLog(){
    char name[50];
    cout<<endl<<"\nEnter the name of the player: ";
    cin>>name;
    cout<<endl<<name<<" scored: ";
    std::fstream logsFile;
    logsFile.open("logs.txt", ios::in);
    std::string logsLine;
    int i=0;
    bool found=false;

    while(getline(logsFile, logsLine)){
        if(i%2==0){
            if(name==logsLine)
            i++;
            found=true;
        }
        else{
            if(found==true){
                cout<<endl<<logsLine;
            }
            found=false;
        }
    }
    cout<<endl<<"in game(s) played";
}

//Function to update logs
void updateLogs(string name, int score){
    fstream LogFile("logs.txt", ios::app|ios::out);
    LogFile<<name;
    LogFile<<score;
}

//function called to extract report from the linked list
void traverseStructure(){
    //display contents of the  second structure
    for(int i=0;i<10;i++){
        cout<<"\nQuestion "<<i+1<<": "<<first->given_question;
        cout<<"\nAnswer: "<<first->correct_answer;
        cout<<"\nYour answer"<<first->given_answer;
        first=first->next;
    }
}
```

```
//function that asks the questions by accessing linked list and saved the report and score
void askQuestion(int n){
   string ans;
   cout<<endl<<n<<". "<<head->question;
   cout<<"\nnow answer it.....";
   cin>>ans;

   SaveData(head->question, head->answer, ans);
   if(ans.compare(head->answer)==0){
      game_score++;
   }
}

//function to play the game
void playGame(){
   char name[50];
   int  turn=1;
   char ch;
   while(true)
         {
                  cout<<"Enter your name: ";
            cin>>name;
   getQuestions();

            for(int i=1;i<=10;i++)
   {
               askQuestion(turn);
            }


            updateLogs(name, game_score);
            cout<<"You have scored: "<<game_score;
            cout<<endl<<"Do you want to see your detailed log? (Y/N)";
            cin>>ch;
            if(ch=='Y'||ch=='y')
               traverseStructure();
            else
               break;
         }
}
//main function
int main(){
   int ch;
//menu is displayed
   while(true){
      cout<<"\n1. Play a new game"<<endl<<"2. View All Logs"<<endl<<"3. View Logs of a
PLayer"<<endl<<"4. Exit"<<endl<<"Enter your choice: ";
      cin>>ch;
//switch statements are used to perform functions based on the choice entered
      switch (ch)
      {
      case 1:
         playGame();
         break;
      case 2:
            showAllLogs();
            break;
        case 3:
            showSpecificLog();
            break;
      case 4:
            exit(0);
        default:
            cout<<"You have entered an invalid choice...."<<endl<<endl;
            break;
        }
    }
}
```

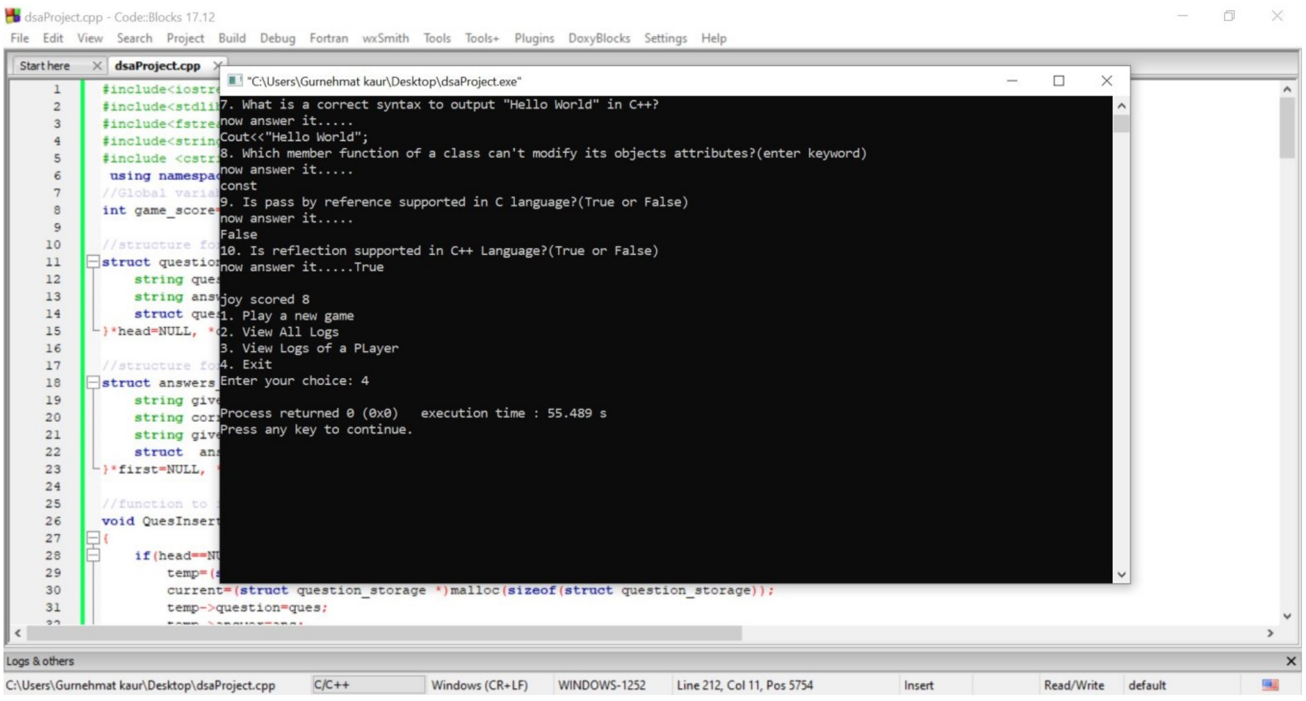
Output:

1. Playing the game


dsaProject.cpp - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
dsaProject.cpp
"C:\Users\Gurnehmat kaur\Desktop\dsaProject.exe"
Logs & others
C:\Users\Gurnehmat kaur\Desktop\dsaProject.cpp
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 132, Col 6, Pos 3536
Insert
Read/Write
default

```
        }
    }
    cout<<endl
}

//Function to
void updateLog
    fstream Lo
    LogFile<<n
    LogFile<<s
}

//function cal
void traverseS
    //display
    for(int i=
        cout<<
        cout<<
        cout<<
        first=
    }
}
//function tha
void askQuesti
    string ans
    cout<<endl
    cout<<"\nn
    cin>>ans;

    SaveData(head->question, head->answer, ans);
    if(ans.compare(head->answer)==0){
```

```
1. Play a new game
2. View All Logs
3. View Logs of a PLayer
4. Exit
Enter your choice: 1

Enter the name of the player: joy


1. Enter the operator which cannot be overloaded.
now answer it.....
::
2. cin is an
now answer it.....
Object
3. The programs machine instructions are store in __ memory segment.
now answer it.....
Idk
4. ios is the base class of istream. Enter True or False.
now answer it.....
True
5. An inline function can execute faster than a normal function. True or False?
now answer it.....
False
6. What is the built in library function to compare two strings?
now answer it.....
Strcmp()
now answer it.....
False
```

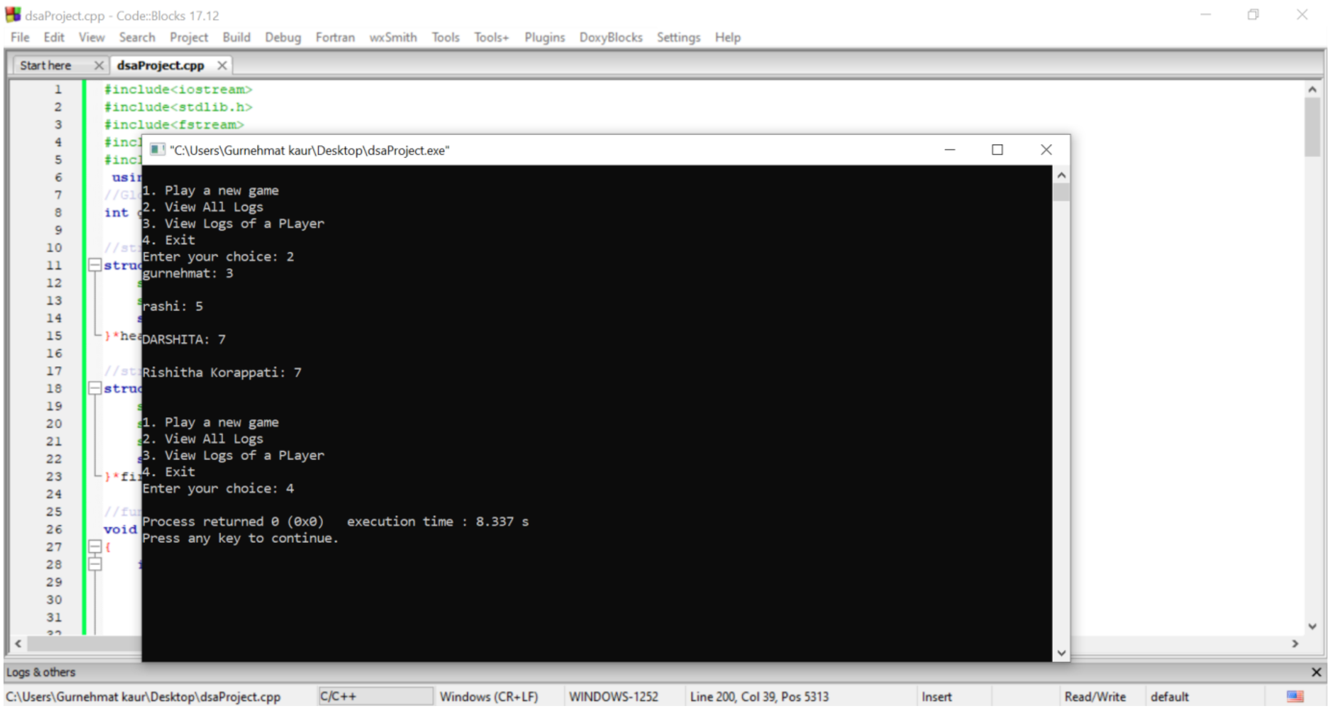

dsaProject.cpp - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
dsaProject.cpp
"C:\Users\Gurnehmat kaur\Desktop\dsaProject.exe"
Logs & others
C:\Users\Gurnehmat kaur\Desktop\dsaProject.cpp
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 212, Col 11, Pos 5754
Insert
Read/Write
default

```
#include<iostr
#include<stdli
#include<fstre
#include<strin
#include <cstri
 using namespa
//Global varia
int game_score

//structure fo
struct questio
    string que
    string ans
    struct que
}*head=NULL, *

//structure fo
struct answers
    string giv
    string cor
    string giv
    struct  an
}*first=NULL,

//function to
void QuesInser
{
    if(head==NU
        temp=(
        current=(struct question_storage *)malloc(sizeof(struct question_storage));
        temp->question=ques;
```

```
7. What is a correct syntax to output "Hello World" in C++?
now answer it.....
Cout<<"Hello World";
8. Which member function of a class can't modify its objects attributes?(enter keyword)
now answer it.....
const
9. Is pass by reference supported in C language?(True or False)
now answer it.....
False
10. Is reflection supported in C++ Language?(True or False)
now answer it.....True

joy scored 8
1. Play a new game
2. View All Logs
3. View Logs of a PLayer
4. Exit
Enter your choice: 4

Process returned 0 (0x0)   execution time : 55.489 s
Press any key to continue.
```

The menu program is displayed and the person is asked for their choice. The choice entered here is 1, which commences a new game. The person's name is taken as input and then all the questions are displayed and their answers are taken as input. Finally, the score is displayed. Here the program has been terminated by choosing option 4 in the next iteration of the menu program.

2. Showing All the Logs

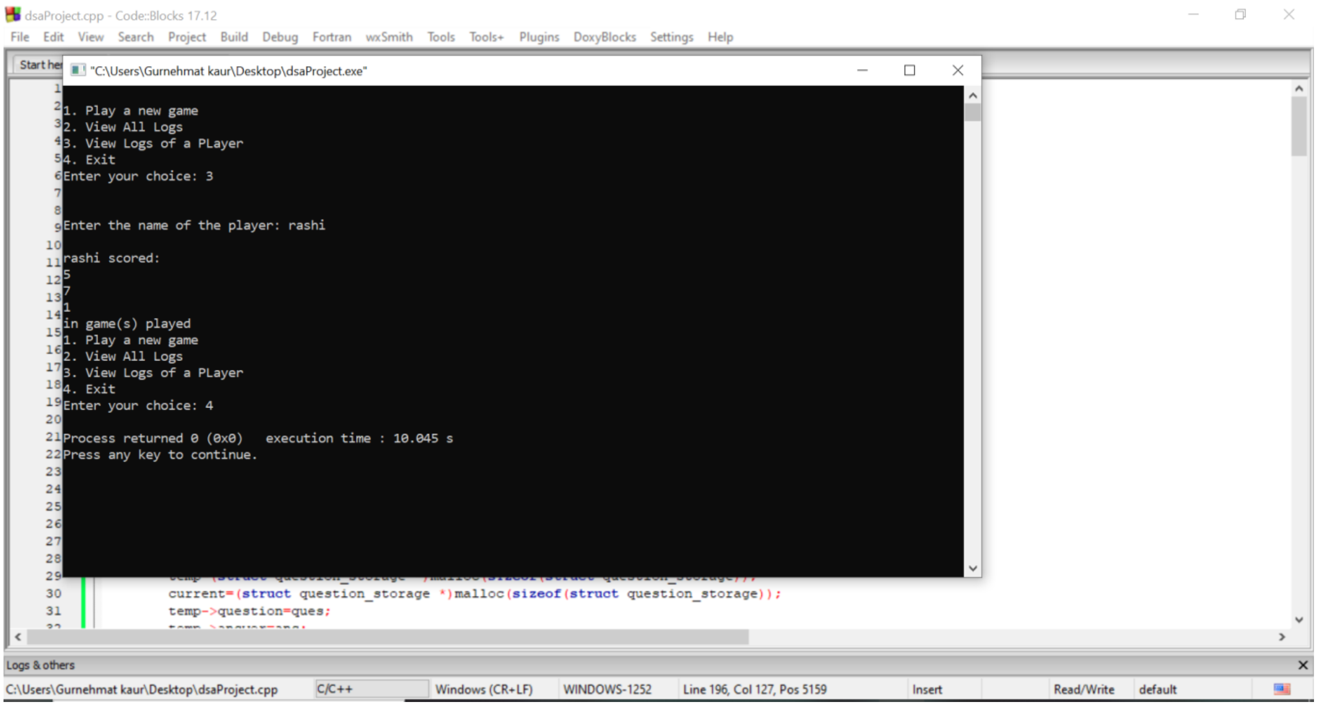

dsaProject.cpp - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
dsaProject.cpp
"C:\Users\Gurnehmat kaur\Desktop\dsaProject.exe"
Logs & others
C:\Users\Gurnehmat kaur\Desktop\dsaProject.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 200, Col 39, Pos 5313 Insert Read/Write default

```
#include<iostream>
#include<stdlib.h>
#include<fstream>
```

```
1. Play a new game
2. View All Logs
3. View Logs of a PLayer
4. Exit
Enter your choice: 2
gurnehmat: 3

rashi: 5

DARSHITA: 7

Rishitha Korappati: 7


1. Play a new game
2. View All Logs
3. View Logs of a PLayer
4. Exit
Enter your choice: 4

Process returned 0 (0x0)   execution time : 8.337 s
Press any key to continue.
```

Here the person has chosen option 2 in the menu program which has retrieved the previous data and displayed it.

3. Showing the logs of a particular person


dsaProject.cpp - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"C:\Users\Gurnehmat kaur\Desktop\dsaProject.exe"
Logs & others
C:\Users\Gurnehmat kaur\Desktop\dsaProject.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 196, Col 127, Pos 5159 Insert Read/Write default

```
1. Play a new game
2. View All Logs
3. View Logs of a PLayer
4. Exit
Enter your choice: 3


Enter the name of the player: rashi

rashi scored:
5
7
1
in game(s) played
1. Play a new game
2. View All Logs
3. View Logs of a PLayer
4. Exit
Enter your choice: 4

Process returned 0 (0x0)   execution time : 10.045 s
Press any key to continue.
```

```
        current=(struct question_storage *)malloc(sizeof(struct question_storage));
        temp->question=ques;
```

Here, the user has selected 3 as the option and looked up the logs of a user named ‘rashi.’ The logs have been accessed and all the scores of ‘rashi’ have been displayed.

VII. Results and discussion

This project is based on the implementation of files, structures, pointers, linked list and queues, during the course of which we gained insight on several ways of implementing these tools. Files are used for performing various functions such as reading, writing, and updating structural data such as structures, class objects etc., and they help store data permanently. The major advantage of files is that they require less memory space for data storage, as opposed to other methods.

Dynamic data structures differ by organization, making performance issues important in selecting the best structure to use. As arrays are static structures, they cannot be easily extended or reduced to fit the data set[9]. To reduce the limitations of arrays another data structure called linked list is used here. Linked list is a linear data structure where each element is a separate object containing some information or data and the address of the next element, which acts as a link to the next object, thus allowing one to access it. They are very useful for adding data dynamically, as memory is allotted when needed and freed after use. This is very useful in performing functions where the amount of data is not known prior to coding the program. The dynamic allotment of data in these structures can be done by using pointers in C++, which is the approach we took. They are also preferred over arrays if the elements in an array tend to change often. This is because it is less costly to add and remove nodes in a Linked List. Many data structures can also be implemented using linked list such as queues, stacks.

Queues are used for enqueueing and dequeuing data similar to a queue of people. They follow first in first out(FIFO) process and functions such as enqueue and dequeue functions are used for removing and adding data into the queue. Elements are always pushed at the back ang taken out from the top. Queues are abstract data structures and can be implemented using arrays, stack, linked list. The most elementary way of implementing a queue is by using and array.

Structures are a key part of our implementation of the linked list and queue models. They are data structures that enable storage of variables of different kinds in a form of a single unit. We have used structures with a self-pointing pointer to simulate linked lists.

VIII. Conclusion

In this project we have been able to utilise concepts learnt during the duration of the course (CSE 2003) in Winter Semester 2019-20 to create a fun and educational project, through a practical and hands-on method. We have gained more insight on the topics of Linked Lists, Queues, Structures and files and on Time Complexity while working on this project.

[9] https://www.cs.cmu.edu/~adamchik/15-121/lectures/Linked%20Lists/linked%20lists.html

IX. References

1. https://www.softwaretestinghelp.com/linked-list/
2. https://www.hackerearth.com/practice/data-structures/linked-list/singly-linked-list/tutorial/
3. https://www.geeksforgeeks.org/data-structures/linked-list/
4. https://www.tutorialspoint.com/cplusplus-program-to-implement-queue-using-linked-list
5. https://www.edureka.co/blog/file-handling-in-cpp/
6. http://www.cplusplus.com/doc/tutorial/files/
7. https://cs.stackexchange.com/
8. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, ‘Introduction to Algorithms’
9. Mark Allen Weiss, ‘Data Structures and Algorithms Analysis in C++’
10. Sumita Arora, ‘Computer Science with c++’