# DSA

# Programming Pathshala

# A training report

Submitted in partial fulfillment of the requirements for the award of degree of

# Bachelor of Technology (Computer Science and Engineering)

**Submitted to**

# LOVELY PROFESSIONAL UNIVERSITY PHAGWARA, PUNJAB



**From 06/01/21 to 07/15/21**

## SUBMITTED BY

**Name of student: Utkarsh Gupta**

**Registration Number: 12017462**

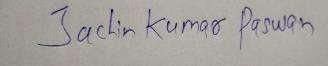
# To whom so ever it may concern

I, **Utkarsh Gupta,12017462** hereby declare that the work done by me on “**DSA**” from **25 May 2022** to **10 July 2022**, is a record of original work done for the partial fulfillment of the requirements for the award of the degree, **DSA(Programming Pathshala).**

**Name of Student: Utkarsh Gupta**

**Registration No.: 12017462**

Sachin Kumar Paswan (12000724)



# ACKNOWLEDGEMENT

I would like to express my gratitude towards my university as well as Programming Pathshala for providing me the golden opportunity to do this wonderful summer training regarding DSA, which also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things. So, I am really thankful to them.

Moreover I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am really thankful to have such a good support of them as they always have my back whenever I need.

Also, I would like to mention the support system and consideration of my parents who have always been there in my life to make me choose right thing and oppose the wrong. Without them I could never had learned and became a person who I am now.

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

**Signature**

**Name:Utkarsh Gupta**

**Registration No.:12017462**

**Sachin Kumar Paswan**

Reg no: 12000724

# Summer Training Certificate



**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page** |
| 1 | Introduction | 6 |
| 2 | Technology learnt | 7-16 |
| 3 | Reason for choosing DSA | 17 |
| 4 | Learning Outcome | 18 |
| 5 | Project & Screenshot | 19-38 |
| 6 | Bibliography | 31-39 |

# INTRODUCTION

* The course name DSA stands for “Data Structures and Algorithms” and Self-paced means, one can join the course anytime. All of the content will be available once one gets enrolled. One can finish it at his own decided speed.
* **Data Structures :** A data structure is a group of data elements that provides the easiest way to store and perform different actions on the data of the computer. A [data structure](https://www.geeksforgeeks.org/data-structures/) is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks. The choice of a good data structure makes it possible to perform a variety of critical operations effectively. An efficient data structure also uses minimum memory space and execution time to process the structure.
* **What is Algorithm?** An algorithm is a finite set of instructions or logic, written in order, to accomplish a certain predefined task. Algorithm is not the complete code or program, it is just the core logic(solution) of a problem, which can be expressed either as an informal high-level description as **pseudocode** or using a **flowchart**
* This course is a complete package that helped me learn Data Structures and Algorithms from basic to an advanced level. The course curriculum has been divided into 8 weeks where one can practice questions & attempt the assessment tests according to his own pace. The course offers me a wealth of programming challenges that will help me to prepare for interviews with top-notch companies like Microsoft, Amazon, Adobe etc.

**TECHNOLOGY LEARNT**

* Learn Data Structures and Algorithms from basic to an advanced level like:
* Learn Topic-wise implementation of different Data Structures & Algorithms as follows

## Analysis of Algorithm

* + - In this I learned about background analysis through a Program and its functions.

## Order of Growth

* + - A mathematical explanation of the growth analysis through limits and functions.
    - A direct way of calculating the order of growth

## Asymptotic Notations

* + - Best, Average and Worst-case explanation through a program.

## Big O Notation

* + - Graphical and mathematical explanation.
    - Calculation
    - Applications at Linear Search

## Omega Notation

* + - Graphical and mathematical explanation.
    - Calculation.

## Theta Notation

* + - Graphical and mathematical explanation.
    - Calculation.

## Analysis of common loops

* + - Single, multiple and nested loops

## Analysis of Recursion

* + - Various calculations through Recursion Tree method

## Space Complexity

* + - Basic Programs
    - Auxiliary Space
    - Space Analysis of Recursion
    - Space Analysis of Fibonacci number

## MATHEMATICS

* + **Finding the number of digits in a number.**

## Arithmetic and Geometric Progressions.

* + **Quadratic Equations.**

## Mean and Median.

* + **Prime Numbers.**

## LCM and HCF

* + **Factorials**

## Permutations and Combinations

* + **Modular Arithmetic BITMAGIC**

## Bitwise Operators in C++

* + - Operation of AND, OR, XOR operators
    - Operation of Left Shift, Right Shift and Bitwise Not

## Bitwise Operators in Java

* + - Operation of AND, OR
    - Operation of Bitwise Not, Left Shift
    - Operation of Right Shift and unsigned Right Shift

## Problem (With Video Solutions): Check Kth bit is set or not

* + - Method 1: Using the left Shift.
    - Method 2: Using the right shift

## RECURSION

* **Introduction to Recursion**

## Applications of Recursion

* **Writing base cases in Recursion**
  + Factorial
  + N-th Fibonacci number

## ARRAYS

* + **Introduction and Advantages**

## Types of Arrays

* + - Fixed-sized array
    - Dynamic-sized array

## Operations on Arrays

* + - Searching
    - Insertions
    - Deletion
    - Arrays v/s other DS
    - Reversing - Explanation with complexity

## SEARCHING

* + **Binary Search Iterative and Recursive**

## Binary Search and various associated problems

* + **Two Pointer Approach Problems SORTING**

## Implementation of C++ STL sort () function in Arrays and Vectors

* + - Time Complexities

## Sorting in Java

* + **Arrays.sort() in Java**

## Collection.sort() in Java

* + **Stability in Sorting Algorithms**
    - Examples of Stable and Unstable Algorithms

## Insertion Sort

* + **Merge Sort**

## Quick Sort

* + - Using Lomuto and Hoare
    - Time and Space analysis
    - Choice of Pivot and Worst case

## Overview of Sorting Algorithms MATRIX

* + **Introduction to Matrix in C++ and Java**

## Multidimensional Matrix

* + **Pass Matrix as Argument**

## Printing matrix in a snake pattern

* + **Transposing a matrix**

## Rotating a Matrix

* + **Check if the element is present in a row and column-wise sorted matrix.**

## Boundary Traversal

* + **Spiral Traversal**

## Matrix Multiplication

* + **Search in row-wise and column-wise Sorted Matrix HASHING**

## Introduction and Time complexity analysis

* **Application of Hashing**

## Discussion on Direct Address Table

* **Working and examples on various Hash Functions**

## Introduction and Various techniques on Collision Handling

* **Chaining and its implementation**

## Open Addressing and its Implementation

* **Chaining V/S Open Addressing**

## Double Hashing

* **C++**
  + Unordered Set
  + Unordered Map

## Java

* + HashSet
  + HashMap

## STRINGS

* + **Discussion of String DS**

## Strings in CPP

* + **Strings in Java**

## Rabin Karp Algorithm

* + **KMP Algorithm LINKED LIST**

## Introduction

* + - Implementation in CPP
    - Implementation in Java
    - Comparison with Array DS

## Doubly Linked List

* + **Circular Linked List**

## Loop Problems

* + - Detecting Loops
    - Detecting loops using Floyd cycle detection
    - Detecting and Removing Loops in Linked List

## STACK

* + **Understanding the Stack data structure**

## Applications of Stack

* + **Implementation of Stack in Array and Linked List**
    - In C++
    - In Java

## QUEUE

* **Introduction and Application**

## Implementation of the queue using array and LinkedList

* + In C++ STL
  + In Java
  + Stack using queue

## DEQUE

* + **Introduction and Application**

## Implementation

* + - In C++ STL
    - In Java

## Problems (With Video Solutions)

* + - Maximums of all subarrays of size k
    - ArrayDeque in Java
    - Design a DS with min max operations

## TREE

* + **Introduction**
    - Tree
    - Application
    - Binary Tree
    - Tree Traversal

## Implementation of:

* + - Inorder Traversal
    - Preorder Traversal
    - Postorder Traversal
    - Level Order Traversal (Line by Line)
    - Tree Traversal in Spiral Form

## BINARY SEARCH TREE

* + **Background, Introduction and Application**

## Implementation of Search in BST

* + **Insertion in BST**

## Deletion in BST

* + **Floor in BST**

## Self-Balancing BST

* + **AVL Tree**

## HEAP

* + **Introduction & Implementation**

## Binary Heap

* + - Insertion
    - Heapify and Extract
    - Decrease Key, Delete and Build Heap

## Heap Sort

* + **Priority Queue in C++**

## Priority Queue I Java GRAPH

* + **Introduction to Graph**

## Graph Representation

* + - Adjacency Matrix
    - Adjacency List in CPP and Java
    - Adjacency Matrix VS List

## Breadth-First Search

* + - Applications

## Depth First Search

* + - Applications

## Shortest Path in Directed Acyclic Graph

* + **Prim's Algorithm/Minimum Spanning Tree**
    - Implementation in CPP
    - Implementation in Java

## Dijkstra's Shortest Path Algorithm

* + - Implementation in CPP
    - Implementation in Java

## Bellman-Ford Shortest Path Algorithm

* + **Kosaraju's Algorithm**

## Articulation Point

* + **Bridges in Graph**

## Tarjan’s Algorithm GREEDY

* **Introduction**

## Activity Selection Problem

* **Fractional Knapsack**

## Job Sequencing Problem BACKTRACKING

* + **Concepts of Backtracking**

## Rat In a Maze

* + **N Queen Problem DYNAMIC PROGRAMMING**

## Introduction

* + **Dynamic Programming**
    - Memoization
    - Tabulation

## TREE

* + **Introduction**
    - Representation
    - Search
    - Insert
    - Delete

## Count Distinct Rows in a Binary Matrix SEGMENT TREE

* **Introduction**

## Construction

* **Range Query**

## Update Query

**DISJOINT SET**

## Introduction

* + **Find and Union Operations**

## Union by Rank

* + **Path Compression**

## Kruskal's Algorithm

* Improved my problem-solving skills by practicing problems to become a stronger developer
  + **Practice problems**
    - This track contains many practice problems for the users which are considered important and must-do as far as Data Structure and Algorithm is concerned
* Developed my analytical skills on Data Structures to use them efficiently
* Solved problems asked in product-based companies’ interviews
* Solved problems in contests similar to coding round for SDE role

**Reason for choosing this technology**

* With advancement and innovation in technology, programming is becoming a highly in-demand skill for Software Developers. Everything you see around yourself from Smart TVs, ACs, Lights, Traffic Signals uses some kind of programming for executing user commands.
* **Data Structures** and **Algorithms** are the identity of a good Software Developer. The interviews for technical roles in some of the tech giants like *Google, Facebook, Amazon, Flipkart* is more focused on measuring the knowledge of Data Structures and Algorithms of the candidates. The main reason behind this is Data Structures and Algorithms improves the problem-solving ability of a candidate to a great extent.
* This course has video lectures of all the topics from which one can easily learn. I prefer learning from video rather than books and notes. I know books and notes and thesis have their own significance but still video lecture or face to face lectures make it easy to understand faster as we are involved practically.
* It has 200+ algorithmic coding problems with video explained solutions.
* It has track based learning and weekly assessment to test my skills.
* It was a great opportunity for me to invest my time in learning instead of wasting it here and there during my summer break in this Covid-19 pandemic.
* This was a lifetime accessible course which I can use to learn even after my training whenever I want to revise

# LEARNING OUTCOMES

Programming is all about data structures and algorithms. Data structures are used to hold data while algorithms are used to solve the problem using that data.

Data structures and algorithms (DSA) goes through solutions to standard problems in detail and gives you an insight into how efficient it is to use each one of them. It also teaches you the science of evaluating the efficiency of an algorithm. This enables you to choose the best of various choices.

For example, you want to search your roll number in 30000 pages of documents, for that you have choices like Linear search, Binary search, etc. So, the more efficient way will be Binary search for searching something in a huge number of data.

So, if you know the DSA, you can solve any problem efficiently. The main use of DSA is to make your code scalable because

* + Time is precious
  + Memory is expensive

# PROJECT

# Noughts and crosses:

# Noughts and Crosses also known as Tic-tac-toe is a simple game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game.

# Game Features:

* Single player mode (you v/s computer) with 3 difficulty levels
* Multi player mode (play against your friend sitting next to you)
* Minimax algorithm (on hard difficulty level)
* Stunning console-like graphics.

# Compilation:

# Just use attached Makefile and type in console make all or in short make. To get rid of object files and executable file type make clean.

# Here Follows the coding Part:-

# The code comprises of Linked list Data Structures and few famous algorithms.

#include "ai.h"

#include "core.h" /\* for prototypes used in minimax() \*/

#include <stdlib.h>

#include <string.h> /\* for memcpy() in minimax() \*/

#include <stdbool.h> /\* for boolean data type in minimax() \*/

int minimax(char node[], int depth, bool maximizingPlayer, int \* move)

{

  int gameResult = anyWinners(node);

  if(depth == 0 || gameResult || fullBoard(node)) /\* depth == 0 or node is a terminal node \*/

    return gameResult \* (depth+1);

  int bestValue, val;

  if (maximizingPlayer)

  {

    bestValue = -11;

    for(int i = 0; i < 9; i++)

    {

      if(node[i] == ' ')

      {

        char child[9];

        memcpy(child, node, 9 \* sizeof(char));

        child[i] = 'X';

        val = minimax(child, depth - 1, false, move);

        if(val > bestValue)

        {

          bestValue = val;

          if(depth == 9) \*move = i+1;

        }

      }

    }

    return bestValue;

  } else {

    bestValue = 11;

    for(int i = 0; i < 9; i++) {

      if(node[i] == ' ')

      {

        char child[9];

        memcpy(child, node, 9 \* sizeof(char));

        child[i] = 'O';

        val = minimax(child, depth - 1, true, move);

        bestValue = bestValue < val ? bestValue : val;

      }

    }

    return bestValue;

  }

}

void aiGenerateMove(int difficultyLevel, int \* field, char board[])

{

  if(difficultyLevel == 1)

  {

    do {

      \*field = rand() % 9 + 1; /\* use seed from drawing() function \*/

    } while(board[\*field-1] != ' ');

  } else if(difficultyLevel == 2) {

    int i;

    // X X -

    // Y Y -

    // Z Z -

    for(i = 0; i < 7; i += 3)

      if(board[i] != ' ' && board[i] == board[i+1] && board[i+2] == ' ')

      {

        \*field = i + 3;

        return;

      }

    // - X X

    // - Y Y

    // - Z Z

    for(i = 1; i < 8; i += 3)

      if(board[i] != ' ' && board[i] == board[i+1] && board[i-1] == ' ')

      {

        \*field = i;

        return;

      }

    // X - X

    // Y - Y

    // Z - Z

    for(i = 0; i < 7; i += 3)

      if(board[i] != ' ' && board[i] == board[i+2] && board[i+1] == ' ')

      {

        \*field = i + 2;

        return;

      }

    // X Y Z

    // X Y Z

    // - - -

    for(i = 0; i < 3; i++)

      if(board[i] != ' ' && board[i] == board[i+3] && board[i+6] == ' ')

      {

        \*field = i + 7;

        return;

      }

    // - - -

    // X Y Z

    // X Y Z

    for(i = 3; i < 6; i++)

      if(board[i] != ' ' && board[i] == board[i+3] && board[i-3] == ' ')

      {

        \*field = i - 2;

        return;

      }

    // X Y Z

    // - - -

    // X Y Z

    for(i = 0; i < 3; i++)

      if(board[i] != ' ' && board[i] == board[i+6] && board[i+3] == ' ')

      {

        \*field = i + 4;

        return;

      }

    // X - -

    // - X -

    // - - -

    if(board[0] != ' ' && board[0] == board[4] && board[8] == ' ')

    {

      \*field = 9;

      return;

    }

    // - - -

    // - X -

    // - - X

    if(board[4] != ' ' && board[4] == board[8] && board[0] == ' ')

    {

      \*field = 1;

      return;

    }

    // X - -

    // - - -

    // - - X

    if(board[0] != ' ' && board[0] == board[8] && board[4] == ' ')

    {

      \*field = 5;

      return;

    }

    // - - X

    // - X -

    // - - -

    if(board[2] != ' ' && board[2] == board[4] && board[6] == ' ')

    {

      \*field = 7;

      return;

    }

    // - - -

    // - X -

    // X - -

    if(board[4] != ' ' && board[4] == board[6] && board[2] == ' ')

    {

      \*field = 3;

      return;

    }

    // - - X

    // - - -

    // X - -

    if(board[2] != ' ' && board[2] == board[6] && board[4] == ' ')

    {

      \*field = 5;

      return;

    }

    aiGenerateMove(1, field, board);

  } else if(difficultyLevel == 3) {

    minimax(board, 9, true, field);

  }

}

#ifndef AI\_H

  #define AI\_H

  #include <stdbool.h> /\* for boolean data type in minimax() \*/

  int minimax(char node[], int depth, bool maximizingPlayer, int \* move);

  void aiGenerateMove(int difficultyLevel, int \* field, char board[]);

#endif

#include "core.h"

#include "view.h" /\* for prototypes used in menuSelection() \*/

#include "game\_modes.h" /\* for prototypes used in chooseMode() \*/

#include <stdio.h>

#include <stdlib.h> /\* for exit() in playAgain() and srand() in drawing() \*/

#include <time.h> /\* for time() in drawing() \*/

#include <stdbool.h> /\* for boolean data type in fullBoard() \*/

void menuSelection(int \* choice, char \* title, char \* options[], int numberOfOptions)

{

  printTitle(title);

  printMenuOptions(options, numberOfOptions);

  printf("   |                                                                        |\n");

  askForMenuNumber(choice, numberOfOptions);

}

void askForMenuNumber(int \* choice, int numberOfOptions)

{

  do {

    printf("   |   Type number:                                                         |\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b");

    while(!scanf("%d", choice)) getchar(); /\* in case you type some letters \*/

  } while(\*choice < 1 || \*choice > numberOfOptions);

}

void chooseMode(void)

{

  int menuChoice;

  char \* numberOfPlayersOptions[] = {

    "1. One player (you vs computer)",

    "2. Two players (you vs friend)"

  };

  menuSelection(&menuChoice, "NUMBER OF PLAYERS", numberOfPlayersOptions, 2);

  menuChoice == 1 ? singleplayer() : multiplayer();

}

void playAgain(void (\*mode)(void))

{

  int menuChoice;

  char \* playAgainOptions[] = {

    "1. Yes",

    "2. No"

  };

  menuSelection(&menuChoice, "DO YOU WANT TO PLAY AGAIN?", playAgainOptions, 2);

  if(menuChoice == 1)

    mode();

  else

    printf("   |                                                                        |\n"

           "   ==========================================================================\n\n");

  exit(EXIT\_SUCCESS);

}

int drawing(void)

{

  srand(time(NULL));

  return rand()%101 < 50 ? 1 : 2;

}

int fullBoard(char board[])

{

  for(int i = 0; i < 9; i++)

    if(board[i] == ' ') return false;

  return true;

}

int whoWon(char ch)

{

  switch(ch)

  {

    case 'O':

    return -1;

    break;

    case 'X':

    return 1;

    break;

  }

}

int anyWinners(char board[])

{

  int i;

  /\* check every row \*/

  for(i = 0; i < 7; i += 3)

    if(board[i] != ' ' && board[i] == board[i+1] && board[i] == board[i+2])

      return whoWon(board[i]);

  /\* check every column \*/

  for(i = 0; i < 3; i++)

    if(board[i] != ' ' && board[i] == board[i+3] && board[i] == board[i+6])

      return whoWon(board[i]);

  /\* check diagonals \*/

  if(board[4] != ' ' && ((board[0] == board[4] && board[0] == board[8]) || (board[2] == board[4] && board[2] == board[6])))

    return whoWon(board[4]);

  return 0;

}

void checkForEndOfGame(int \* i, int \* field, char board[], void (\*mode)(void))

{

  // 5 symbols or more can cause an end of the game (note that we iterate i from 0)

  // if any of the 9 moves didn't end the game with winner then it is a tie

  if(\*i > 3 && anyWinners(board))

  {

    printGameboard(board);

    printf("   |                                \"%c\" WINS!                               |\n", board[\*field-1]);

    playAgain(mode);

  } else if(\*i == 8) {

    printGameboard(board);

    printf("   |                              IT'S A TIE!                               |\n");

    playAgain(mode);

  }

}

#ifndef CORE\_H

  #define CORE\_H

  void menuSelection(int \* choice, char \* title, char \* options[], int numberOfOptions);

  void askForMenuNumber(int \* choice, int numberOfOptions);

  void chooseMode(void);

  void playAgain(void (\*mode)(void));

  int drawing(void);

  int fullBoard(char board[]);

  int whoWon(char ch);

  int anyWinners(char board[]);

  void checkForEndOfGame(int \* i, int \* field, char board[], void (\*mode)(void));

#endif

#include "game\_modes.h"

#include "core.h"

#include "view.h"

#include "ai.h" /\* for aiGenerateMove() prototype \*/

#include <stdio.h>

void multiplayer(void)

{

  int startingPlayer = drawing();

  printTitle("INSTRUCTIONS");

  printf("   |   Let's flip a coin:                                                   |\n"

         "   |   PLAYER %d starts the game!                                            |\n", startingPlayer);

  printGameboard("123456789");

  int field;

    char board[9] = "         "; /\* nine spaces, one for each field \*/

  for(int i = 0; 1; i++)

  {

    askForMenuNumber(&field, 9);

    /\* ask for symbol again if given field is not empty \*/

    if(board[field-1] != ' ')

    {

      i--;

      continue;

    }

    board[field-1] = i % 2 ? 'O' : 'X';

    printTitle("GAMEBOARD");

    checkForEndOfGame(&i, &field, board, multiplayer);

    /\* prepare view for the next iteration \*/

    int whoseTurnIsIt = (startingPlayer + i) % 2 + 1;

    printf("   |   PLAYER %d it's your turn now, make a move!                            |\n", whoseTurnIsIt);

    printGameboard(board);

  }

}

void singleplayer(void)

{

  /\* choose difficulty level \*/

  int difficultyLevel;

  char \* difficultyLevelOptions[] = {

    "1. Easy",

    "2. Normal",

    "3. Hard"

  };

  menuSelection(&difficultyLevel, "DIFFICULTY LEVEL", difficultyLevelOptions, 3);

  /\* decide who starts the game and print instructions \*/

  int startingPlayer = drawing();

  char player[2][78] = {

    "   |   YOU start the game!                                                  |",

    "   |   COMPUTER starts the game!                                            |"

  };

  printTitle("INSTRUCTIONS");

  printf("   |   Let's flip a coin:                                                   |\n"

         "%s\n", player[startingPlayer-1]);

  printGameboard("123456789");

  int field;

    char board[9] = "         "; /\* nine spaces, one for each field \*/

  for(int i = 0; 1; i++)

  {

    int human = (startingPlayer + i) % 2;

    /\* depending on player ask for sign or run AI \*/

    if(human)

    {

      askForMenuNumber(&field, 9);

      /\* ask for symbol again if given field is not empty \*/

      if(board[field-1] != ' ')

      {

        i--;

        continue;

      }

    } else {

      waitingForMove();

      aiGenerateMove(difficultyLevel, &field, board);

      printf("   |   Type number: %d                                                       |\n", field);

    }

    if(difficultyLevel == 3 && startingPlayer == 1)

      board[field-1] = i % 2 ? 'X' : 'O'; /\* you play as a O on hard level because minimax is always X \*/

    else

      board[field-1] = i % 2 ? 'O' : 'X';

    printTitle("GAMEBOARD");

    checkForEndOfGame(&i, &field, board, singleplayer);

    /\* prepare view for the next iteration \*/

    if(human)

      printf("   |   COMPUTER is making a move!                                           |\n");

    else

      printf("   |   YOUR turn now, make a move!                                          |\n");

    printGameboard(board);

  }

}

#ifndef GAME\_MODES\_H

  #define GAME\_MODES\_H

  void multiplayer(void);

  void singleplayer(void);

#endif

#include <stdio.h>

#include "view.h" /\* for printLogo() prototype \*/

#include "core.h" /\* for menuSelection() and chooseMode() prototype \*/

int main(void)

{

  printLogo();

  int menuChoice;

  char \* mainMenuOptions[] = {

    "1. Play",

    "2. Quit"

  };

  menuSelection(&menuChoice, "MAIN MENU", mainMenuOptions, 2);

  if(menuChoice == 1)

    chooseMode();

  else

    printf("   |                                                                        |\n"

           "   ==========================================================================\n\n");

  return 0;

}

#include "view.h"

#include <stdio.h>

#include <unistd.h> /\* for sleep() in waitingForMove() \*/

#include <string.h> /\* for strlen() in printTitle() and printMenuOptions() \*/

void printLogo(void)

{

  /\* ASCII Generator http://www.network-science.de/ascii/ (font: kban) \*/

  printf("\n"

         "   ==========================================================================\n"

         "   |                                                                        |\n"

         "   |                                                                        |\n"

         "   |                                                                        |\n"

         "   |        '|.   '|'                          '||        .                 |\n"

         "   |         |'|   |    ...   ... ...    ... .  || ..   .||.   ....         |\n"

         "   |         | '|. |  .|  '|.  ||  ||   || ||   ||' ||   ||   ||. '         |\n"

         "   |         |   |||  ||   ||  ||  ||    |''    ||  ||   ||   . '|..        |\n"

         "   |        .|.   '|   '|..|'  '|..'|.  '||||. .||. ||.  '|.' |'..|'        |\n"

         "   |                                   .|....'                              |\n"

         "   |                                                                        |\n"

         "   |                                             '||                        |\n"

         "   |                        ....   .. ...      .. ||                        |\n"

         "   |                       '' .||   ||  ||   .'  '||                        |\n"

         "   |                       .|' ||   ||  ||   |.   ||                        |\n"

         "   |                       '|..'|' .||. ||.  '|..'||.                       |\n"

         "   |                                                                        |\n"

         "   |                                                                        |\n"

         "   |           ..|'''.|                                                     |\n"

         "   |         .|'     '  ... ..    ...    ....   ....    ....   ....         |\n"

         "   |         ||          ||' '' .|  '|. ||. '  ||. '  .|...|| ||. '         |\n"

         "   |         '|.      .  ||     ||   || . '|.. . '|.. ||      . '|..        |\n"

         "   |          ''|....'  .||.     '|..|' |'..|' |'..|'  '|...' |'..|'        |\n"

         "   |                                                                        |\n"

         "   |                                                                        |\n");

}

void printGameboard(char board[])

{

  printf("   |                                                                        |\n"

         "   |                                 |     |                                |\n"

         "   |                              %c  |  %c  |  %c                             |\n"

         "   |                            \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_                           |\n"

         "   |                                 |     |                                |\n"

         "   |                              %c  |  %c  |  %c                             |\n"

         "   |                            \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_                           |\n"

         "   |                                 |     |                                |\n"

         "   |                              %c  |  %c  |  %c                             |\n"

         "   |                                 |     |                                |\n"

         "   |                                                                        |\n", board[0], board[1], board[2], board[3], board[4], board[5], board[6], board[7], board[8]);

}

void printTitle(char \* title)

{

  printf("   |                                                                        |\n"

         "   ==========================================================================\n"

         "   | :::: %s ", title);

  for(int i = 0; i < 64 - strlen(title); i++)

    printf(":");

  printf(" |\n"

         "   ==========================================================================\n"

         "   |                                                                        |\n");

}

void printMenuOptions(char \* options[], int numberOfOptions)

{

  for(int i = 0; i < numberOfOptions; i++)

  {

    printf("   |   %s", options[i]);

    for(int j = 0; j < 69 - strlen(options[i]); j++)

      printf(" ");

    printf("|\n");

  }

}

void waitingForMove(void)

{

  printf("   |   Thinking.                                                            |\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b");

  for(int i = 0; i < 2; i++)

  {

    fflush(stdout);

    sleep(1);

    printf(".");

  }

  printf("\n");

}

#ifndef VIEW\_H

  #define VIEW\_H

  void printLogo(void);

  void printGameboard(char board[]);

  void printTitle(char \* title);

  void printMenuOptions(char \* options[], int numberOfOptions);

  void waitingForMove(void);

#endif

# Screenshot and Working Output:-

# ==========================================================================

# | |

# | |

# | |

# | '|. '|' '|| . |

# | |'| | ... ... ... ... . || .. .||. .... |

# | | '|. | .| '|. || || || || ||' || || ||. ' |

# | | ||| || || || || |'' || || || . '|.. |

# | .|. '| '|..|' '|..'|. '||||. .||. ||. '|.' |'..|' |

# | .|....' |

# | |

# | '|| |

# | .... .. ... .. || |

# | '' .|| || || .' '|| |

# | .|' || || || |. || |

# | '|..'|' .||. ||. '|..'||. |

# | |

# | |

# | ..|'''.| |

# | .|' ' ... .. ... .... .... .... .... |

# | || ||' '' .| '|. ||. ' ||. ' .|...|| ||. ' |

# | '|. . || || || . '|.. . '|.. || . '|.. |

# | ''|....' .||. '|..|' |'..|' |'..|' '|...' |'..|' |

# | |

# | |

# | |

# ==========================================================================

# | :::: MAIN MENU ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | 1. Play |

# | 2. Quit |

# | |

# | Type number: 1 |

# | |

# ==========================================================================

# | :::: NUMBER OF PLAYERS ::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | 1. One player (you vs computer) |

# | 2. Two players (you vs friend) |

# | |

# | Type number: 1 |

# | |

# ==========================================================================

# | :::: DIFFICULTY LEVEL :::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | 1. Easy |

# | 2. Normal |

# | 3. Hard |

# | |

# | Type number: 2 |

# | |

# ==========================================================================

# | :::: INSTRUCTIONS :::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | Let's flip a coin: |

# | YOU start the game! |

# | |

# | | | |

# | 1 | 2 | 3 |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | 4 | 5 | 6 |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | 7 | 8 | 9 |

# | | | |

# | |

# | Type number: 5 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | COMPUTER is making a move! |

# | |

# | | | |

# | | | |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | X | |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | | |

# | | | |

# | |

# | Thinking... |

# | Type number: 3 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | YOUR turn now, make a move! |

# | |

# | | | |

# | | | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | X | |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | | |

# | | | |

# | |

# | Type number: 9 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | COMPUTER is making a move! |

# | |

# | | | |

# | | | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | X | |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | | X |

# | | | |

# | |

# | Thinking... |

# | Type number: 1 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | YOUR turn now, make a move! |

# | |

# | | | |

# | O | | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | X | |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | | X |

# | | | |

# | |

# | Type number: 2 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | COMPUTER is making a move! |

# | |

# | | | |

# | O | X | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | X | |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | | X |

# | | | |

# | |

# | Thinking... |

# | Type number: 8 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | YOUR turn now, make a move! |

# | |

# | | | |

# | O | X | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | X | |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | O | X |

# | | | |

# | |

# | Type number: 6 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | COMPUTER is making a move! |

# | |

# | | | |

# | O | X | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | X | X |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | O | X |

# | | | |

# | |

# | Thinking... |

# | Type number: 4 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | YOUR turn now, make a move! |

# | |

# | | | |

# | O | X | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | O | X | X |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | | O | X |

# | | | |

# | |

# | Type number: 77 |

# | Type number: 7 |

# | |

# ==========================================================================

# | :::: GAMEBOARD ::::::::::::::::::::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | |

# | | | |

# | O | X | O |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | O | X | X |

# | \_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_ |

# | | | |

# | X | O | X |

# | | | |

# | |

# | IT'S A TIE! |

# | |

# ==========================================================================

# | :::: DO YOU WANT TO PLAY AGAIN? :::::::::::::::::::::::::::::::::::::: |

# ==========================================================================

# | |

# | 1. Yes |

# | 2. No |

# | |

# | Type number: 2 |

# | |

# ==========================================================================

**Biblography**

* Google
* VS code
* Greeks for Greek