A) Team Members:

1. Naay Balodia: 140020106 (Team Leader)
2. Kapil Vaidya: 140050006
3. Pintu Raj: 140020086
4. Lalitprakash Chauhan: 140020071

B) Introduction:

1) One of the most popular ways of recreation in today's digital world is computer games.

2) With these thoughts in our mind we decided to use our limited programming knowledge to create the famous computer board games
   a) "MINESWEEPER"- a regular feature of Windows Operating System.
   b) 7 UP 7 DOWN

3) We have also made Sudoku auto solve which can solve a Sudoku puzzle given by user.

4) We also decided to link it with ACCOUNT MANAGEMENT SYSTEM wherein player would have to first login and place bet before playing the game.

C) Problem Statement:

To create a GAMING ARCADE with Minesweeper as one of the games provided by Windows Operating System. The arcade would also include “7 up 7 down”.

D) About The Minesweeper Game:

The player is initially presented with a grid of undistinguished squares. Some randomly selected squares, unknown to the player, are designated to contain mines. Typically, the size of the grid and the number of mines are set in advance by the user, either by entering the numbers or selecting from defined skill levels depending on the implementation. The game is played by revealing squares of the grid, typically by clicking them with a mouse. If a square containing a mine is revealed, the player loses the game. Otherwise, a digit is revealed in the square, indicating the number of adjacent squares (typically, out of the possible eight) that contain mines. In typical implementations, if this number is zero then the square appears blank, and the surrounding squares are automatically also revealed. By using logic, the player can in many instances use this information to deduce that certain other squares are mine-free, in which case they may be safely revealed, or mine-filled, in which they can be marked as such (which, in typical implementations, is effected by right-clicking the square and indicated by a flag graphic).The player successfully completes the game when he uncovers the whole grid correctly or flags all the mines present correctly.

E) About Sudoku game: (source Wikipedia)

It is a logic based combinatorial number placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid (also called "boxes", "blocks", "regions", or "sub-squares") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a unique solution.

F) About 7up 7down:

It is a game based on pure luck. Two dices are rolled. Player decides whether the sum of numbers appearing on the dices would be less than 7, greater than 7 or equal to 7. The player wins if he/she makes the correct prediction

G)Basic Algorithm of Minesweeper:

Concepts of object oriented programming have been used. A “class Game” has been constructed. There are four arrays declared as private members along with several other private variables and private member functions. Some public member functions have also been declared:

The various private members of the class are:

i) **a[10][10]:** It would store the location of randomly generated mines.
ii) **print[10][10]:** It would store location of the squares uncovered by the player
iii) **flag[10][10]:** It would store location of squares flagged as mine by the player
iv) **n_mines[10][10]:** It would store number of mines surrounding a particular square in the grid
v) **nr,nc,n:** They store number of rows, columns and mines in the grid respectively

vi) **x,y:** They record the coordinates of mouse click in pixels
vii) **row,column:** They store the index of row and column selected by the player
viii) **check,check1:** They continuously keep a check of whether the player has won or lost the game
ix) **bet:** It stores the value of bet points placed by user
x) **ac:** It stores the details of player account with which the player is logged in

The various private member functions of the class are:

i) **void processEvents():** It keeps check of various inputs provided by user such as left click, right click and calls uncover(), flagit(), unflagit() functions accordingly. It also keeps a check of whether user has won the game with the help of check_win() function.
ii) **void render():** It renders graphical display of minesweeper board on the screen. It also displays win or lose messages and gives corresponding sound effects at the end of the game.
iii) **void get_location():** It initialises value of row and column of the square selected by the player by clicking
iv) **int uncover():** It checks whether the square uncovered by player has mine or not. It uncovers the square selected by the player. It also uses help of uncover1() function which recursively calls itself in case no mine surrounds the square selected by the player.
v) **void flagit():** It flags the square selected by the user
vi) **void unflagit():** It unflags a flagged square by the player
vii) **int check_win():** It keeps a check of whether the player has won the game or not.

The various public member functions of the class are:

i) **init_array():** It initialises all the elements of a[10][10], flag[10][10] and print[10][10] to 0. Some of the elements of a[10][10] are initialised to 1 where randomly generated mines are placed. It also initialises the array n_mines[10][10] with number of mines surrounding each square in the grid.

ii) **init_level():** It asks the player to select the level he/she wishes to play and accordingly sets the value of nr, nc and n. It also calls the withdraw() function which returns it the bet amount placed by the player after deducting it from player's account.

iii) **run():** It begins the game of minesweeper by opening a new window with graphical display and mouse interface by using the help of processEvents() and render() functions**.** If the player wins the game it credits the player's account with the points won with the help of deposit_win() function.

H)Basic Algorithm for Account Management System:

A class account has been declared which has following:

Private VARIABLES:

i) int acno: To store the account number
ii) char name[50]: To store name of the account holder
iii) int deposit: To store money deposited in the account
iv) char pass[20]: To store password of the account holder

Public MEMBER FUNCTIONS:

i) void create_account(): function to get data from user
ii) void show_account(): function to show data on screen
iii) void modify(): function to modify new data
iv) void modify_pass(char[]): function to modify password
v) void dep(int): function to accept amount and add to balance

amount

vi) void draw(int): function to accept amount and subtract from balance amount

vii) void report(): function to show data in tabular format

viii) int retacno(): function to return account number

ix) int retdeposit(): function to return balance amount

x) int checkpass(char[]): function to verify password

Following functions are declared outside the class:

i) void write_account(): function to write record in binary file

ii) void display_sp(int): function to display account details given by user

iii) void modify_account(int): function to modify record of file

iv) void modify_account_pass(char[]): function to modify account password

v) void delete_account(int): function to delete record of file

vi) void display_all(): function to display all account details

vii) void deposit(int): function to desposit amount for given account

viii) void deposit_win(int,int): function to deposit the win amount for particular game

ix) int withdraw(int): function to withdraw amount for given account. It return the bet amount place by the user

I) Basic Algorithm for 7 up 7 down:

Concepts of object oriented programming have been used. A “class Game1” has been constructed which has following:

Private VARIABLES:

i) i,j: They store the values of randomly generated numbers in the dice.

ii) x: Store x coordinate of position of mouse click.

iii) m: records whether sum of numbers appearing is less than 7, greater than 7 or equal to 7.

iv) check: It is initialized to -1. Stores 0 if player loses the game or 1 if player wins the game.
v) check7: Stores 1 if player wins the game by correctly guessing the sum as 7 otherwise stores 0.
vi) bet: Stores the bet amount placed by the player
vii) ac: It stores the details of player account with which the player is logged in

Private MEMBER FUNCTIONS:

i) process(): It keeps check of various inputs provided by user such as click. It accordingly sets the value of x. It then assigns random values to i and j and initializes m. It then checks whether user has won or lost by comparing the values of x and m.

ii) render1(): It renders graphical display of 7 up 7 down board on the screen. It also displays win or lose messages and gives corresponding sound effects at the end of the game.

Public MEMBER FUNCTIONS:

i) run1(): It initializes bet with help of withdraw() function. It then creates a new game window. The game runs with the help of process() and render1() functions until plyer wins or loses.

J) Basic Algorithm of Sudoku Autosolve:

The algorithm used for Sudoku auto solve is backtracking. The following functions are declared:

i) int grid(int i, int j): Checks and returns in which sub grid i and j belong.
ii) int validate(int val,int b[][9],int i,int j): Checks if b[i][j] can have the value as “val” by comparing with entries in corresponding row, column and grid.
iii) void func(int a[][9],int b[][9],int i,int j,int val,int &check):

Calls itself recursively and prints the solution of the sudoku puzzle.

K)Things that can be done:

More games can be added to the gaming arcade to make it more interesting.