Description:

- The program is a chess game.
- It is played by 2 players.
- Player 1 is white & Player 2 is Black.
- You win the game when the next move is to capture your opponent king and he can't save himself.

Design:

- The board is a 2D array of 8x8.
- We made the black squares='+' & the white squares='-'.
- The capital letters pieces are black like 'P, Q, B, ….'
- The small letters pieces are white like 'p, q, b, ….'
- The order of rows begins from 0 to 7 in the code
  But in the board begins from 1 to 8.
- The order of columns begins from 0 to 7 in the code
  But in the board begins from A to H.

Design/implementation assumptions & necessary details:

- We made the board global and another 2D array "Test"
  Like the board 8x8 To implement a move and see it's valid or not if it's not valid it undo the move.
- The player who begins the game is always white.
- If he entered small letters it prints ("Please enter Capital letters and scan again.
- If the player is white and played with black piece it prints ("This is not white Piece.") and scan again and same with black.

- If he entered a position that there is no piece in it, it prints ("There is no piece in this position.").

- If any player's pawn has reached the end of the board it's promotion so it prints ("It's promotion. Here's a list you can promote the pawn to (r , n, b, q). The piece you chose is:") and you can change the pawn to queen or knight or bishop or rook.

Data structures:

- int make_ch_num (char a)
- void print_board ()
- int self_check1(int x, int l, int y, int m)
- int self_check2(int x, int l, int y, int m)
- void pawn1(int x, int l, int y, int m) //WHITE
- void pawn2(int x, int l, int y, int m) //BLACK
- void knight1(int x, int l, int y, int m) //white
- void knight2(int x, int l, int y, int m) //Black
- void rook1(int x, int l, int y, int m) //White
- void rook2(int x, int l, int y, int m) //Black
- void bishop1(int x, int l, int y, int m) //WHITE
- void bishop2(int x, int l, int y, int m) //BLACK
- void queen1(int x, int l, int y, int m) //White
- void queen2(int x, int l, int y, int m) //Black
- void king1(int x, int l, int y, int m) //WHITE
- void king2(int x, int l, int y, int m) //BLACK
- int check1(int x, int l, int y, int m)
- int check2(int x, int l, int y, int m)

- int save_king2(int x, int l, int y, int m)
- int save_king1(int x, int l, int y, int m)
- int help_me2(int x, int l, int y, int m)
- int help_me1(int x, int l, int y, int m)

Important functions/modules:

- pawn:
  If the pawn in the first state, you can move one or two square forwards
  If it's in general state it can move one square forwards (if the position is empty)
  The pawn is in the end of the board it can be converted to any other piece
  The pawn capture diagonal in forward.
- Bishop:
  if the player gives a move which the current row subtract the given row is equal the current column subtract the given column it's a valid move and the bishop moves diagonal.
  If there's any piece (black or white) in the way of the given position it's a not valid move.
- Check:
  Search where is the position of the king of the player and see if the opponent's pieces can reach the king or not
  If it can its check.
- Save king
  When the player's king in check it sees whether the next move can break the check
  if it is it's a valid move
  if it is not it's a not valid move.

Pseudo code: (we call the functions in the main)

- Pawn:

```
int pawn1 (int x, int l, int y, int m) {//WHITE
        If p in basic case then
                if one move forward or 2 moves forward and same
                column and empty given square then
                        right ← 1
                end if
        end if
        If one move forward and same column and empty given
        square then
                Right ← 1
        end if

        If the given move contains black piece then

                If moves diagonal forward then
                        lose_from_black[countb] ← BW[y][m];
                end if
        end if
        If right = 0 then
                not_valid ← 1
        else if right = 1 then
                not_valid ← 0
        else if right = 2 then
                not_valid ← 2
        end if
return not_valid
}
```

- Check:

```
int check1(int x, int l, int y, int m) {//check to player 2
    z ← test[y][m]
    u ← test[x][l]
    test[y][m] ← test[x][l]
    test[x][l] ← BW1[x][l]
    for I from 0 to 8
        for j from 0 to 8
                if find the black king then
                        get ← 1
                        break
                end if
        end for
        if get = 1 then
                break
        end if
    end for
    for t from 0 to 8
        for h from 0 to 8
                if test[t][h] = 'p' then
                        if pawn can capture king then
                                check ← 1
                                break
                        end if
                else if test[t][h] = 'n' then
                        if knight can capture king then
                                check ← 1
                                break
                        end if
```

```
            else if test[t][h] = 'r' then
                if rook can capture king then
                    check ← 1
                    break
                end if
            else if test[t][h] = 'b' then
                If bishop can capture king then
                    check ← 1
                    break
                end if
            else if test[t][h] = 'q' then
                If queen can capture king then
                    check ← 1
                    break
                end if
            else if test[t][h] = 'k' then
                If king can capture black king then
                    check ← 1
                    break
                end if
            end if

    return check

    }
```

User Manual:

- Give 4 givens which is (the current column, the current row, the column you want to move to, the raw you want to move to) example: A7A6.
- Player1 is white & Player2 is black.

Sample runs:

Pawn:

E7E6, E2E3, A7A5, H2H4.

Rook:

A7A5, H1H3, A8A6, H3F3.

Bishop:

D7D5, D2D4, C8H3

Queen:

D8G5, D1F3

Knight:

G1F3

Check:

E7E5, G1H3, D8H4, A2A3, F8C5, B2B3, H4F2