Basic feature: “Undo one move” for each player

This feature, I implemented for both: game against computer or in the game of two players. After each move from the player, the program asks if you want to undo your last move? You enter 'U' (without the quotes), otherwise enter 'C' (without the quotes).
Important note: command U and C only accept these characters in uppercase. If you entered U then you should enter the number of moves you want undo.

This feature work details:

For games against computer if you enter number x then the gameboard will be the same as x move before. As in example below, after entering number U 2, you gameboard recovered to the gameboard which was before you first move.

```
C:\Users\erzas\Documents\Problem-Solving Project1>a
How many human players [1-2]?
1
=========
|7||8||9|
|4||5||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
5
=========
|7||8||9|
|4||O||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
C
Computer places the mark:
=========
|X||8||9|
|4||O||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
9
=========
|X||8||O|
|4||O||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
U
Please enter, how many moves you want to undo? [1, 2]
2
=========
|7||8||9|
|4||5||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
```

Another example, in below in case, if you entered 1 instead of 2, you gameboard changed so you last move was undo.

```
C:\Users\erzas\Documents\Problem-Solving Project1>a
How many human players [1-2]?
1
=========
|7||8||9|
|4||5||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
5
=========
|7||8||9|
|4||O||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
C
Computer places the mark:
=========
|X||8||9|
|4||O||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
9
=========
|X||8||O|
|4||O||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
U
Please enter, how many moves you want to undo? [1, 2]
1
=========
|X||8||9|
|4||O||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
```

For game with 2 players, I count moves by both players.

```
=========
Player 1, please place your mark [1-9]:
5
=========
|7||8||9|
|4||O||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
C
=========
|7||8||9|
|4||O||6|
|1||2||3|
=========
Player 2, please place your mark [1-9]:
7
=========
|X||8||9|
|4||O||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
C
=========
|X||8||9|
|4||O||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
9
=========
|X||8||O|
|4||O||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
U
Please enter, how many moves you want to undo? [1, 3]
2
=========
|7||8||9|
|4||O||6|
|1||2||3|
=========
Player 2, please place your mark [1-9]:
```

As it showed in example higher, after entering U and then 2, the game undo the last move by both players, so the turn changed from 1 player to 2, and the gameboard was the same as before the last move from 2 player.

Implementation details:

I developed this feature, by using a 3D array, so I saved every state of the gameboard, and I also created new variable moves which contain information about how many moves were done. In pictures below, you can see my implementation.

```
int games[10][3][3] = {};
int currentPlayer = 1;
int numOfHumanPlayers;
int moves = 0;

while(1) {
    printf("How many human players [1-2]?\n");
    scanf("%d", &numOfHumanPlayers);
    if(numOfHumanPlayers >= 1 && numOfHumanPlayers <= 2)
        break;
    puts("Incorrect number of human players, please try again");
}

while(1) {
    if(isFull(games[moves])) {
        break;
    }
    ++moves;
    for (int i = 0; i < 3; i ++)
        for (int j = 0; j < 3; ++j)
            games[moves][i][j] = games[moves-1][i][j];
    if(currentPlayer <= numOfHumanPlayers) {
        printGameBoard(games[moves]);
    }
    if(currentPlayer == 1) {
        printf("Player 1, please place your mark [1-9]:\n");
        placeMarkByHumanPlayer(games[moves], CIRCLE); // 1 player move
    } else if(numOfHumanPlayers == 2) {
        printf("Player 2, please place your mark [1-9]:\n");
        placeMarkByHumanPlayer(games[moves], CROSS); // 2 player move
    } else {
        printf("Computer places the mark:\n"); //so 2 player is Computer
        placeMarkByComputerPlayer(games[moves]);
    }
    if(currentPlayer <= numOfHumanPlayers) {
        printGameBoard(games[moves]);
        puts("If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)");
        int cnt = 0;
        char Undo[101];
```

In picture up, you can see that I have games[k][i][j], where k is gameboard after k moves were done(including computer moves), i and j are coordinates on board.

```
    int cnt = 0;
    char Undo[101];
    do {
        if(++cnt > 1) {
            printf("You last command was incorrect, please enter 'U' (without the quotes) if you want to undo move/moves, otherwise enter 'C' (without the quotes)\n");
        }
        scanf("%s", Undo);
    }while(strlen(Undo) > 1 || Undo[0] != 'C' && Undo[0] != 'U');

    if(Undo[0] == 'U') {
        cnt = 0;

        if(numOfHumanPlayers == 2) {
            printf("Please enter, how many moves you want to undo? [1, %d]\n", moves);
            int skip;
            do {
                if(++cnt > 1) {
                    printf("You last command was incorrect, please enter number of moves you want to undo within interval [1, %d]\n", moves);
                }
                scanf("%d", &skip);
            } while(skip < 1 || skip > moves);

            if(!(skip & 1)) {
                currentPlayer = 3 - currentPlayer;
            }
            moves -= skip;
        } else {
            printf("Please enter, how many moves you want to undo? [1, %d]\n", (moves + 1) / 2); //since almost half of moves was done by computer
            int skip;
            do {
                if(++cnt > 1) {
                    printf("You last command was incorrect, please enter number of moves you want to undo within interval [1, %d]\n", (moves+1)/2);
                }
                scanf("%d", &skip);
            } while(skip < 1 || skip > (moves+1)/2);

            moves -= 2*skip-1;
        }
        continue;
    }
```

If you are playing against a computer, it can be seen that almost half of moves was done by computer.
Also, after undoing x moves if x is even, next move will be done by other player.

Challenge Feature: Improved Computer AI strategies

Computer AI, in my tic tac toe after improving, moves in the most optimal way.
As you know, in a tic tac toe game, it is impossible to win if you didn't make any mistake. In a game against my AI, you can't win even if you move optimally the result will be at best a draw. During the game if you made at least two mistakes, the computer will win.

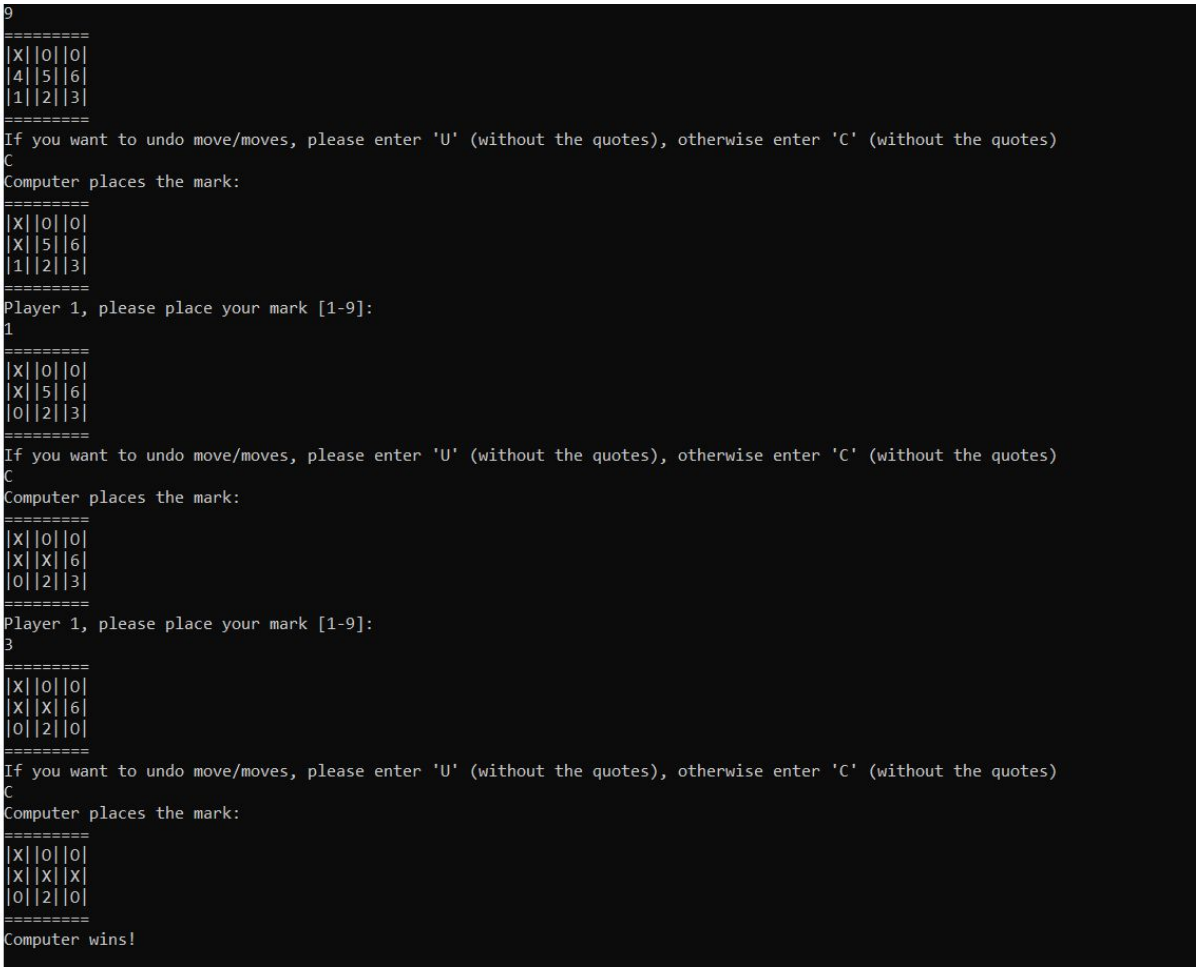
This feature Implementation details:

So to develop such AI, I used Game Theory and Dynamic Programming. In particular, I used Nim Game Theory and Dynamic Programming over subsets.
So let's save to subsets A and B, where Circle's is subset A and Cross's is subset B. From set A, B we look for moves to other subsets where we can win(we can win if other players will lose), if no such set we look for subsets where we can draw, if no such either we will lose. So losing states is when we have vertical, horizontal or diagonal lines of size 3, so before moving we look at any of that. If the opposite player won the game in the previous move, otherwise we look for the optimal move for us.
So in order to not calculate every state many times, I used a dp[][] array to save state for which we previously calculated the results and move[][] to save which move was optimal during such state. Pictures of implementation you can see below:
dp[][] = -1(if we didn't calculate for such state)
dp[][] = 1(if we win in this state)
dp[][] = 0(if we lose in this state)
dp[][] = 2(if we draw in this state)

```
                *res = nxt;
                move[maskA][maskB] = id+1;
            }
        }
    }

    if(*res == 1) *res = 0;

    return *res;
}

void fillDP() {
    for (int i = 0; i < (1<<9); ++i)
        for (int j = 0; j < (1<<9); ++j)
            dp[i][j] = -1;
}

int placeMarkByComputerPlayer(int gameBoard[3][3]) {
    int A = 0, B = 0;
    for (int i = 0; i < 3; ++i)
        for(int j = 0; j < 3; ++j) {
            if(gameBoard[i][j] == CIRCLE) A |= (1<<(posId(i,j)-1));
            if(gameBoard[i][j] == CROSS) B |= (1<<(posId(i,j)-1));
        }

    CalcDp(B, A);
    int id = move[B][A];

    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if(posId(i, j) == id) {
                gameBoard[i][j] = CROSS;
                return 1;
            }
        }
    }

    return 0;
}

int main() {
    fillDP();
```

Sample Runs against Computer:

```
C:\Users\erzas\Documents\Problem-Solving Project1>a
How many human players [1-2]?
1
=========
|7||8||9|
|4||5||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
8
=========
|7||O||9|
|4||5||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
C
Computer places the mark:
=========
|X||O||9|
|4||5||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
9
=========
|X||O||O|
|4||5||6|
|1||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
C
Computer places the mark:
=========
|X||O||O|
|X||5||6|
|1||2||3|
=========
Player 1, please place your mark [1-9]:
1
=========
|X||O||O|
|X||5||6|
|O||2||3|
=========
If you want to undo move/moves, please enter 'U' (without the quotes), otherwise enter 'C' (without the quotes)
C
Computer places the mark:
=========
```



As you can see in the sample run up, I didn't move optimally in my first 2 moves, so AI found the winning strategy, and I have no chance after.

Reference List:

Nim game Theory - https://en.wikipedia.org/wiki/Nim#:~:text=Nim%20is%20a%20mathematical%20game,from%20distinct%20heaps%20or%20piles.&text=This%20is%20called%20normal%20play,way%20that%20Nim%20is%20played.

Dynamic Programming over subsets - https://codeforces.com/blog/entry/337?locale=en