## 1. Input validation

### Introduction

All user input is expected to be single integers. This feature will check and invalidate all unexpected input.

### Details

All input is obtained by this getinput() function which return the valid input.

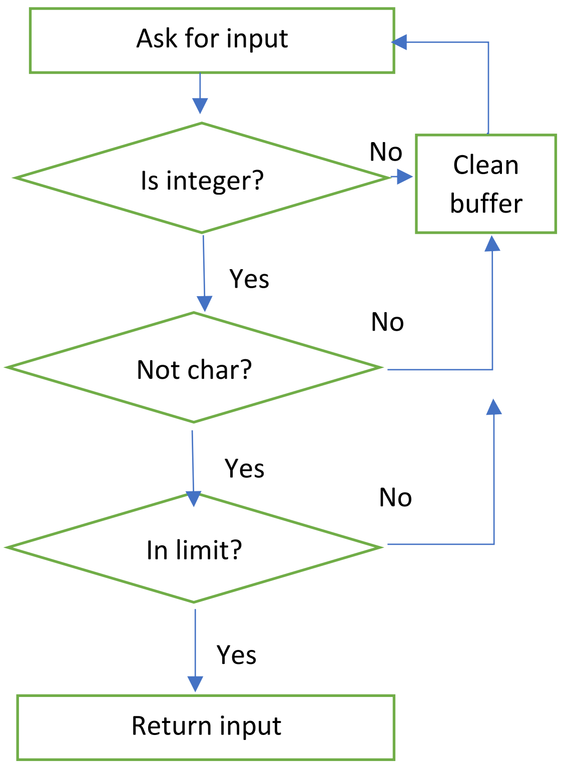
The function requires two argument, the upper and lower limit of the desired input, it then runs a simple while loop until a valid input is returned.

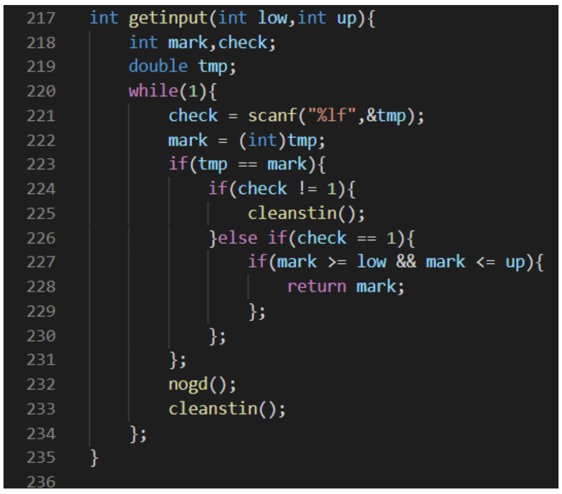
Firstly, it scans and store the input and the return value of scanf. It then check if the input is a single integer without decimal by converting the input to integer and seeing if they are equal. For characters, they will be stored as integers, so the next part will check the returned value of scanf.

If the returned value is 1, that means it successfully reads an integer, then it will check is the number is within the upper bound and lower bound, if that's also true, it will return the input.

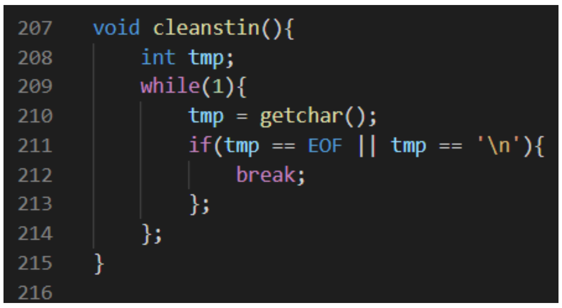
However, if the returned value is not 1, that means the user input is not a number, which is a invalid input, then the function cleanstin() will be called as to avoid infinite looping when the user entered a character. cleanstin() scan and store the character in standard input to a unused holder until the End of line or a new line which remain in the input buffer.

Lastly, the loop will ask for user input again until a valid one is entered.





```c
int getinput(int low,int up){
    int mark,check;
    double tmp;
    while(1){
        check = scanf("%lf",&tmp);
        mark = (int)tmp;
        if(tmp == mark){
            if(check != 1){
                cleanstin();
            }else if(check == 1){
                if(mark >= low && mark <= up){
                    return mark;
                };
            };
        };
        nogd();
        cleanstin();
    };
}
```



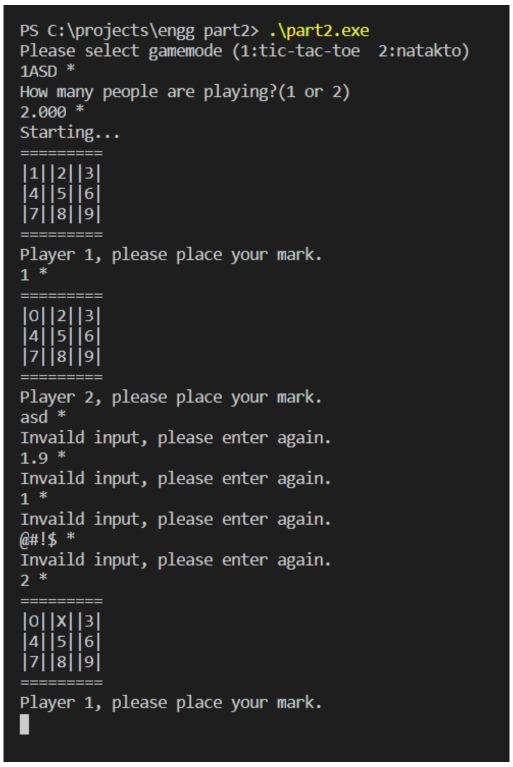
```c
void cleanstin(){
    int tmp;
    while(1){
        tmp = getchar();
        if(tmp == EOF || tmp == '\n'){
            break;
        };
    };
}
```

When the user is in a game, the input is further validate by the checkmark() function.

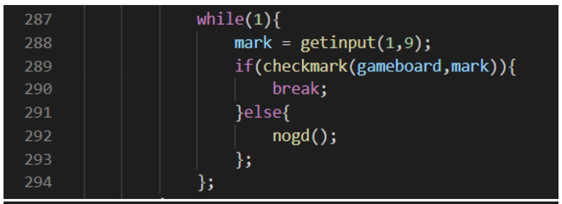
After the input is validated by getinput(),the checkmark() function will check if the mark placed by the user is occupied or not. Simply, if the place of the mark on the gameboard is empty, it returns 1 to break the while loop, if not, then it returns 0 to continue asking for input.

Note that when a valid input is followed by other invalid inputs (like 1A or 1@_asd), the valid input is read and the whole input is considered valid. Moreover, valid numbers with all decimal places being 0 (like 1.0) are also valid.

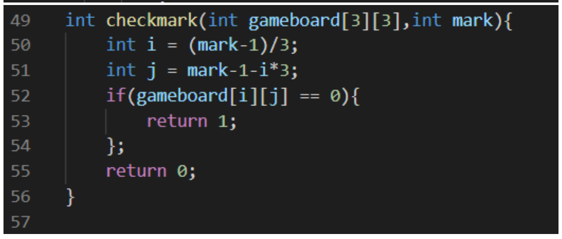
**Sample runs (* at the end is user input)**



```
PS C:\projects\engg part2> .\part2.exe
Please select gamemode (1:tic-tac-toe  2:natakto)
1ASD *
How many people are playing?(1 or 2)
2.000 *
Starting...
=========
|1||2||3|
|4||5||6|
|7||8||9|
=========
Player 1, please place your mark.
1 *
=========
|O||2||3|
|4||5||6|
|7||8||9|
=========
Player 2, please place your mark.
asd *
Invaild input, please enter again.
1.9 *
Invaild input, please enter again.
1 *
Invaild input, please enter again.
@#!$ *
Invaild input, please enter again.
2 *
=========
|O||X||3|
|4||5||6|
|7||8||9|
=========
Player 1, please place your mark.
█
```

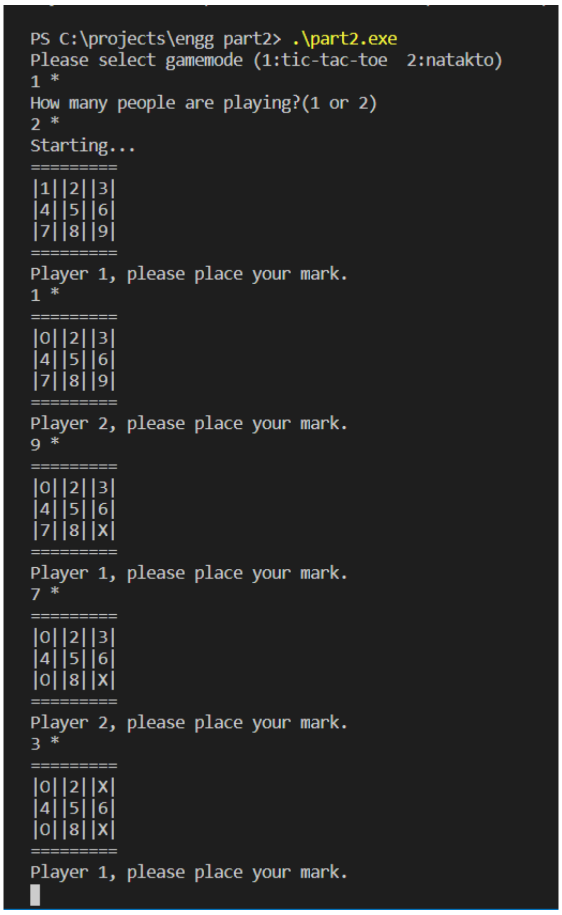


```
        while(1){
            mark = getinput(1,9);
            if(checkmark(gameboard,mark)){
                break;
            }else{
                nogd();
            };
        };
```



```
int checkmark(int gameboard[3][3],int mark){
    int i = (mark-1)/3;
    int j = mark-1-i*3;
    if(gameboard[i][j] == 0){
        return 1;
    };
    return 0;
}
```

**Sample runs**



```
PS C:\projects\engg part2> .\part2.exe
Please select gamemode (1:tic-tac-toe  2:natakto)
1 *
How many people are playing?(1 or 2)
2 *
Starting...
=========
|1||2||3|
|4||5||6|
|7||8||9|
=========
Player 1, please place your mark.
1 *
=========
|O||2||3|
|4||5||6|
|7||8||9|
=========
Player 2, please place your mark.
9 *
=========
|O||2||3|
|4||5||6|
|7||8||X|
=========
Player 1, please place your mark.
7 *
=========
|O||2||3|
|4||5||6|
|O||8||X|
=========
Player 2, please place your mark.
3 *
=========
|O||2||X|
|4||5||6|
|O||8||X|
=========
Player 1, please place your mark.
█
```

**References**

1.https://stackoverflow.com/questions/53056369/c-infinite-loop-when-char-input-instead-of-int

## 2. Additional game mode

### Introduction

Another game mode called Notakto is added, it is similar to tic-tac-toe, however both player play as CROSS and the player who get 3 in a row loses. Unlike to tic-tac-toe, this game never draws.

### Details

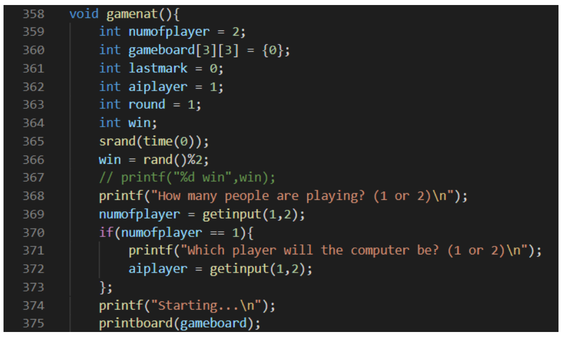
The whole game is in a function called gamenat(). Firstly, the function initiates the gameboard, set all place to empty and initiate other variables to keep track of the number of players, player of the AI, the current round and the pervious mark placed. Then it will ask for the number of people playing, and, if there is only one, which player will the AI be.

The flow is simple, the while loop will loop until there is a winner. First ask the user to place the mark using the getmark() function. The function will validate and return the valid mark. If there is only one player and it is the AI's round, the function will call the ailogic() or ailose() function and return the mark placed by the AI.

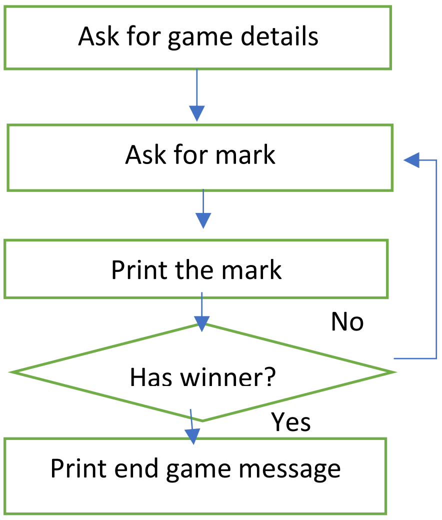
After obtaining the mark placed, it is printed to the gameboard using the printboard() function, which is the same with that in part 1.

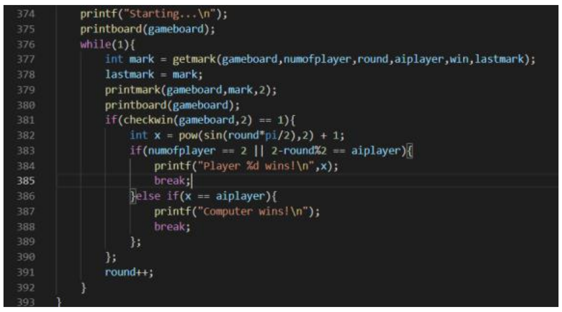
Then the haswinner() function, the same function in part 1, is called to check if anyone has scored 3 in a row. If the function returns 1, that means there is a winner, the winning message is printed with the last player as winner.

The current player is determined by 2 – (round)mod2, the mod2 of round gives if the round is odd or even, 2 minus it will give the current player. The last player is determined by a squared sine function + 1 which oscillate between 1 and 2 according to round.

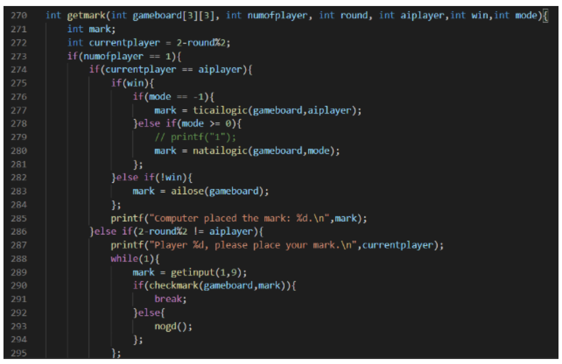


```c
void gamenat(){
    int numofplayer = 2;
    int gameboard[3][3] = {0};
    int lastmark = 0;
    int aiplayer = 1;
    int round = 1;
    int win;
    srand(time(0));
    win = rand()%2;
    // printf("%d win",win);
    printf("How many people are playing? (1 or 2)\n");
    numofplayer = getinput(1,2);
    if(numofplayer == 1){
        printf("Which player will the computer be? (1 or 2)\n");
        aiplayer = getinput(1,2);
    };
    printf("Starting...\n");
    printboard(gameboard);
```

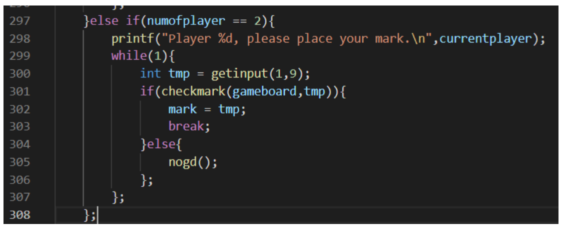




```c
    printf("Starting...\n");
    printboard(gameboard);
    while(1){
        int mark = getmark(gameboard,numofplayer,round,aiplayer,win,lastmark);
        lastmark = mark;
        printmark(gameboard,mark,2);
        printboard(gameboard);
        if(checkwin(gameboard,2) == 1){
            int x = pow(sin(round*pi/2),2) + 1;
            if(numofplayer == 2 || 2-round%2 == aiplayer){
                printf("Player %d wins!\n",x);
                break;
            }else if(x == aiplayer){
                printf("Computer wins!\n");
                break;
            };
        };
        round++;
    }
}
```



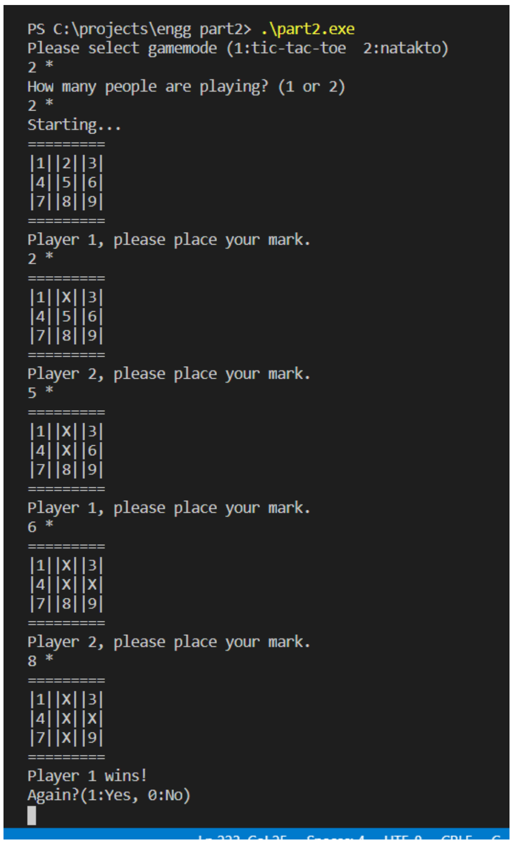
```c
int getmark(int gameboard[3][3], int numofplayer, int round, int aiplayer,int win,int mode){
    int mark;
    int currentplayer = 2-round%2;
    if(numofplayer == 1){
        if(currentplayer == aiplayer){
            if(win){
                if(mode == -1){
                    mark = ticailogic(gameboard,aiplayer);
                }else if(mode >= 0){
                    // printf("1");
                    mark = natailogic(gameboard,mode);
                };
            }else if(!win){
                mark = ailose(gameboard);
            };
            printf("Computer placed the mark: %d.\n",mark);
        }else if(2-round%2 != aiplayer){
            printf("Player %d, please place your mark.\n",currentplayer);
            while(1){
                mark = getinput(1,9);
                if(checkmark(gameboard,mark)){
                    break;
                }else{
                    nogd();
                };
            };
```



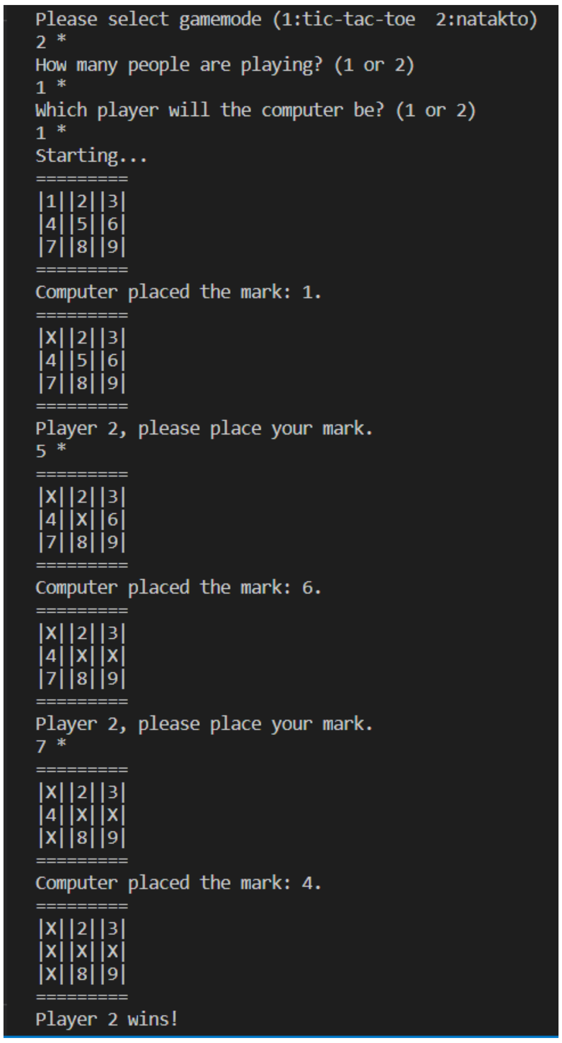
```c
    }else if(numofplayer == 2){
        printf("Player %d, please place your mark.\n",currentplayer);
        while(1){
            int tmp = getinput(1,9);
            if(checkmark(gameboard,tmp)){
                mark = tmp;
                break;
            }else{
                nogd();
            };
        };
    };
```

The getmark() function ^

## Sample runs (* at the end is user input)



```
PS C:\projects\engg part2> .\part2.exe
Please select gamemode (1:tic-tac-toe  2:natakto)
2 *
How many people are playing? (1 or 2)
2 *
Starting...
=========
|1||2||3|
|4||5||6|
|7||8||9|
=========
Player 1, please place your mark.
2 *
=========
|1||X||3|
|4||5||6|
|7||8||9|
=========
Player 2, please place your mark.
5 *
=========
|1||X||3|
|4||X||6|
|7||8||9|
=========
Player 1, please place your mark.
6 *
=========
|1||X||3|
|4||X||X|
|7||8||9|
=========
Player 2, please place your mark.
8 *
=========
|1||X||3|
|4||X||X|
|7||X||9|
=========
Player 1 wins!
Again?(1:Yes, 0:No)
```



```
Please select gamemode (1:tic-tac-toe  2:natakto)
2 *
How many people are playing? (1 or 2)
1 *
Which player will the computer be? (1 or 2)
1 *
Starting...
=========
|1||2||3|
|4||5||6|
|7||8||9|
=========
Computer placed the mark: 1.
=========
|X||2||3|
|4||5||6|
|7||8||9|
=========
Player 2, please place your mark.
5 *
=========
|X||2||3|
|4||X||6|
|7||8||9|
=========
Computer placed the mark: 6.
=========
|X||2||3|
|4||X||X|
|7||8||9|
=========
Player 2, please place your mark.
7 *
=========
|X||2||3|
|4||X||X|
|X||8||9|
=========
Computer placed the mark: 4.
=========
|X||2||3|
|X||X||X|
|X||8||9|
=========
Player 2 wins!
```

## 3. Improved AI

**Introduction**

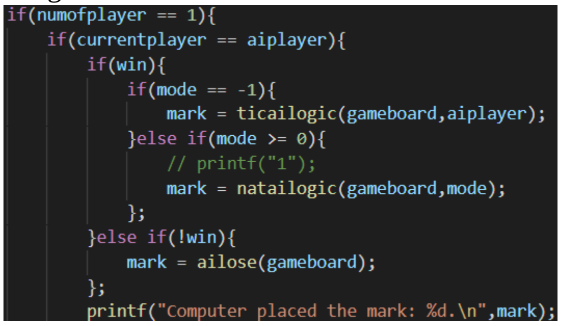
Advanced AI is created for both game mode, the AI is governed by three logic, two for the game modes and one for random AI. The two logic plays optimally and, by the nature of these game, will never lose when playing as Player 1.

**Details**

General

The AI can play as both player 1 and player 2, the human will be answer to choose what the AI will play as after answering 1 people is playing.

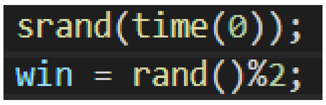
When getmark() is called and the current round is the AI's round, the three logic for the AI will be called, depending on the win variable. If it is 0, ailose() will be called; If it is 1, the logic for the game mode will be called.



```
if(numofplayer == 1){
    if(currentplayer == aiplayer){
        if(win){
            if(mode == -1){
                mark = ticailogic(gameboard,aiplayer);
            }else if(mode >= 0){
                // printf("1");
                mark = natailogic(gameboard,mode);
            };
        }else if(!win){
            mark = ailose(gameboard);
        };
        printf("Computer placed the mark: %d.\n",mark);
```

### 1. The ailose() function

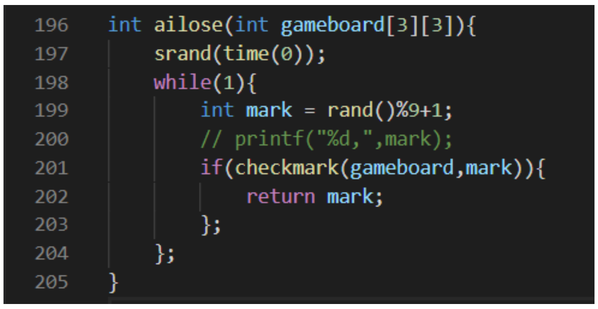
At the start of the game, a random number is generated using rand() and the system time as the seed, if it is even, the AI will use this function.



```
srand(time(0));
win = rand()%2;
```

This function is simple, the AI generate a random mark, if the place is empty on the gameboard (using the checkmark function), it will return this mark.

It is quite difficult to lose to this AI.

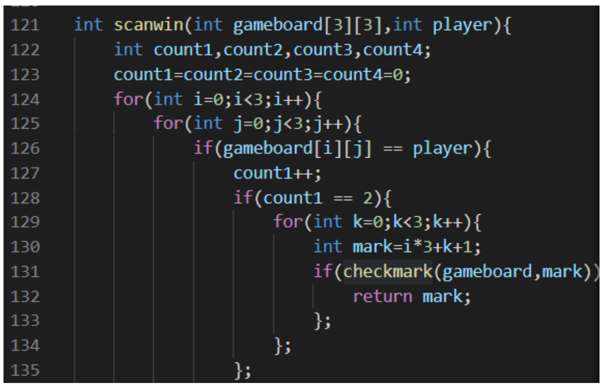


```
int ailose(int gameboard[3][3]){
    srand(time(0));
    while(1){
        int mark = rand()%9+1;
        // printf("%d,",mark);
        if(checkmark(gameboard,mark)){
            return mark;
        };
    };
}
```

## 2. AI logic for tic-tac-toe

The logic for tic-tac-toe is named ticailogic(). The logic has 4 rules.

Firstly, the scanwin() function is used to scan for two in a roll for the aiplayer, so that the AI would take the win when given the chance.



```
int scanwin(int gameboard[3][3],int player){
    int count1,count2,count3,count4;
    count1=count2=count3=count4=0;
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(gameboard[i][j] == player){
                count1++;
                if(count1 == 2){
                    for(int k=0;k<3;k++){
                        int mark=i*3+k+1;
                        if(checkmark(gameboard,mark))
                            return mark;
                        };
                    };
                };
```

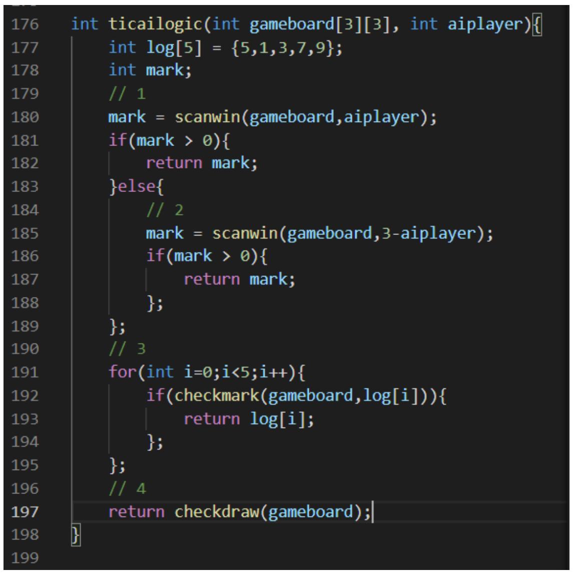
The function would search the gameboard horizontally, vertically, diagonally and anti-diagonally for the aiplayer's mark, if there is already two mark and the remaining place is empty, the function will return the remaining place. On the left is the scanwin() function in the horizonal direction.

Next, the same function will be used to scan for the human player's mark, so that AI will stop the human's win when given the chance.

If no one is winning, the AI would return one of the five mark hardcoded in the array, whichever is empty on the gameboard, the five marks is the center and the four concern.

Finally, if none of the above applies, the AI would check the checkdraw() function, which will return the first available place.

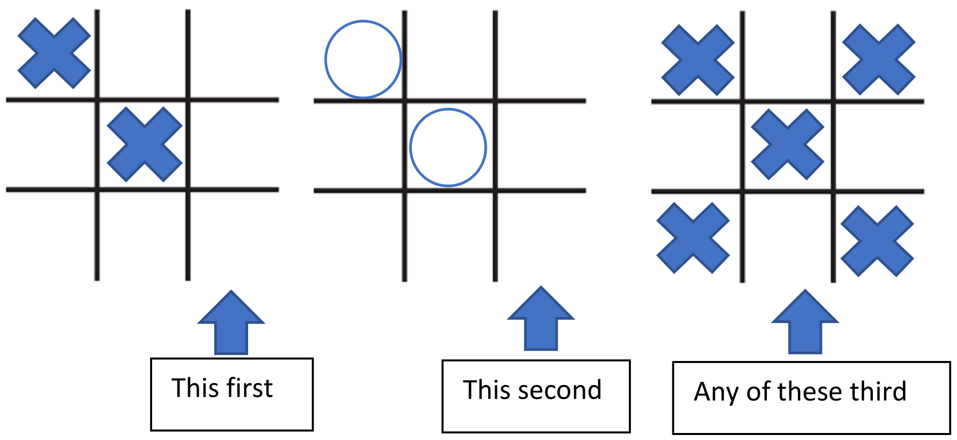
These four logics guarantees that the AI will not lose, only draw or win.



```
int ticailogic(int gameboard[3][3], int aiplayer){
    int log[5] = {5,1,3,7,9};
    int mark;
    // 1
    mark = scanwin(gameboard,aiplayer);
    if(mark > 0){
        return mark;
    }else{
        // 2
        mark = scanwin(gameboard,3-aiplayer);
        if(mark > 0){
            return mark;
        };
    };
    // 3
    for(int i=0;i<5;i++){
        if(checkmark(gameboard,log[i])){
            return log[i];
        };
    };
    // 4
    return checkdraw(gameboard);
}
```

Logic flow

**Winning First → Prevent Losing second → Priority place Third → Any remaining place**



## 3. AI logic for Notakto

The logic for AI placing the mark in Notakto is to find the marks NOT to place. The marks NOT to place is stored in an array called bad[].

Firstly, priority is given to the center ( 5 ), if it is unoccupied, the AI will place 5.

(step 1)

Secondly, if the last mark (placed by the human player) is an even number (i.e. 2,4,6,8), all even numbered place is marked. Then, the last mark and both the row and column containing respect to the center is also marked. (for example, the opposite position of 1 is 9)

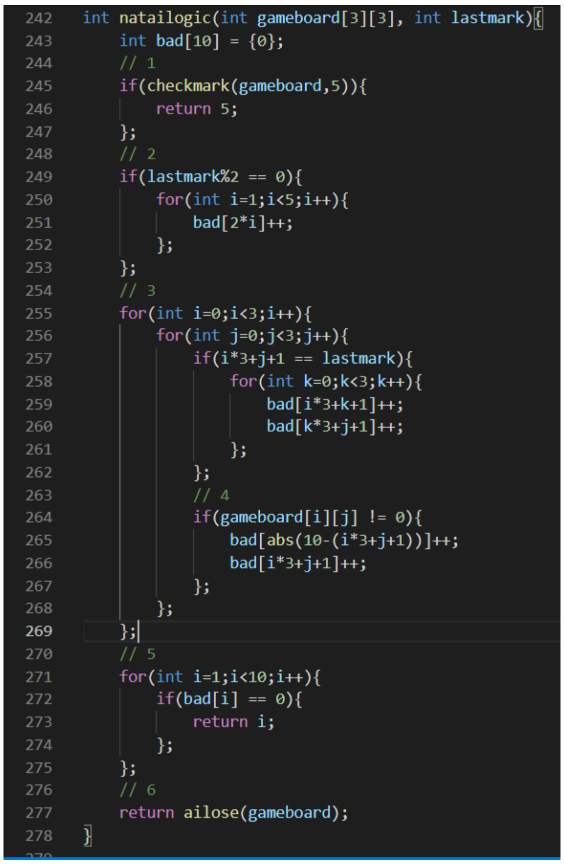
(step 2 - 4)

Then any first place that is not marked is returned.

(step 5).

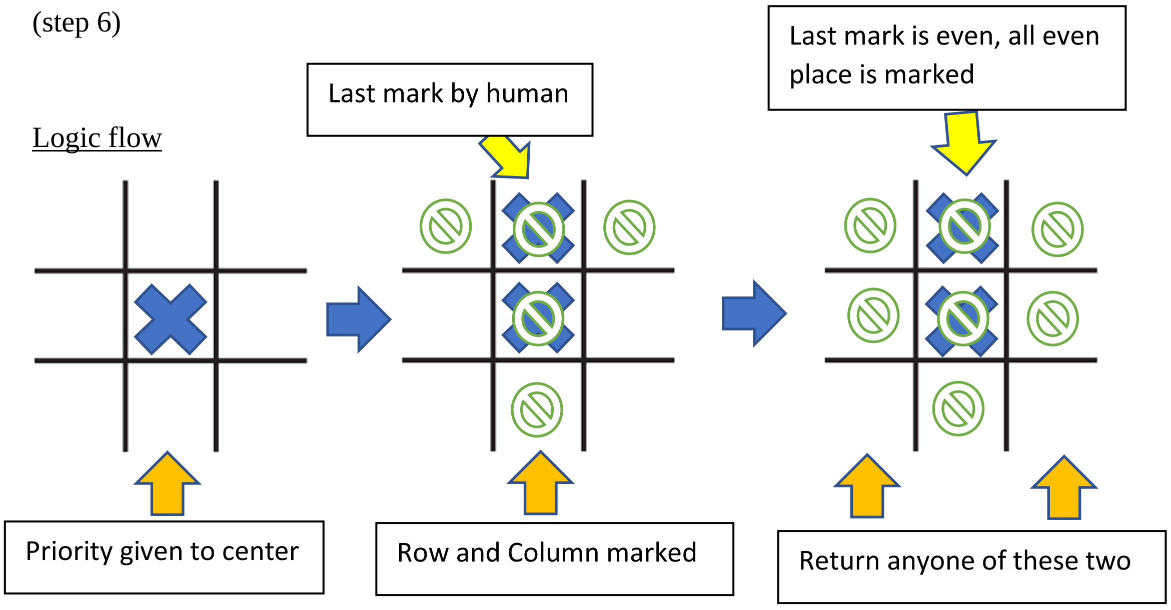
If all the places is marked NOT to place, a random unoccupied place will be returned using the ailose() function.

(step 6)



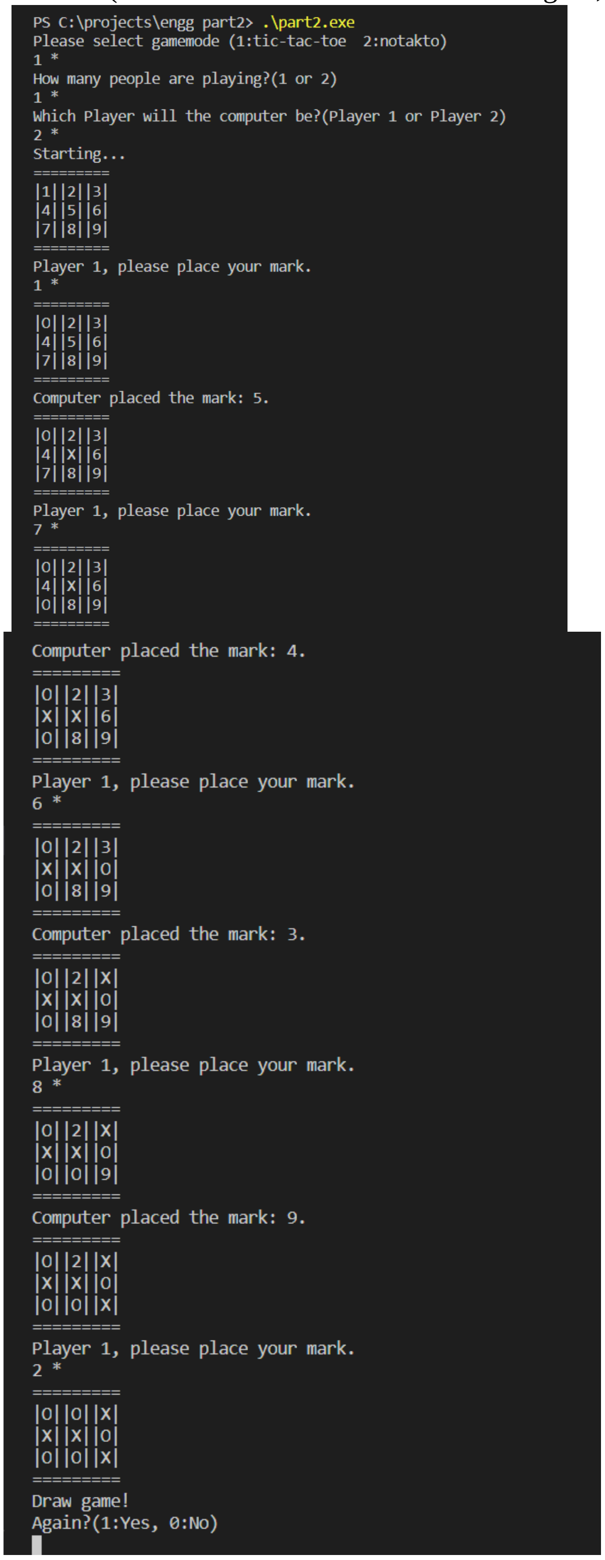
```c
int natailogic(int gameboard[3][3], int lastmark){
    int bad[10] = {0};
    // 1
    if(checkmark(gameboard,5)){
        return 5;
    };
    // 2
    if(lastmark%2 == 0){
        for(int i=1;i<5;i++){
            bad[2*i]++;
        };
    };
    // 3
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(i*3+j+1 == lastmark){
                for(int k=0;k<3;k++){
                    bad[i*3+k+1]++;
                    bad[k*3+j+1]++;
                };
            };
            // 4
            if(gameboard[i][j] != 0){
                bad[abs(10-(i*3+j+1))]++;
                bad[i*3+j+1]++;
            };
        };
    };
    // 5
    for(int i=1;i<10;i++){
        if(bad[i] == 0){
            return i;
        };
    };
    // 6
    return ailose(gameboard);
}
```

Logic flow

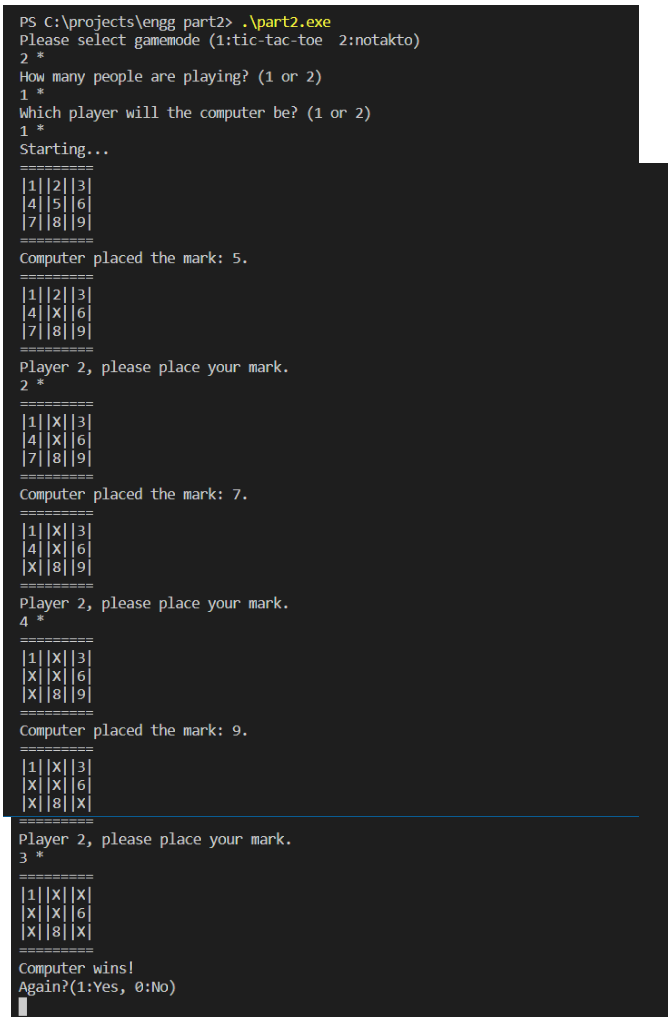

Last mark by human
Last mark is even, all even place is marked
Priority given to center
Row and Column marked
Return anyone of these two

## <u>Sample runs(* at the end is user input)</u>

(uncomment line 286 to test for losing AI, line 285 for other two AI)

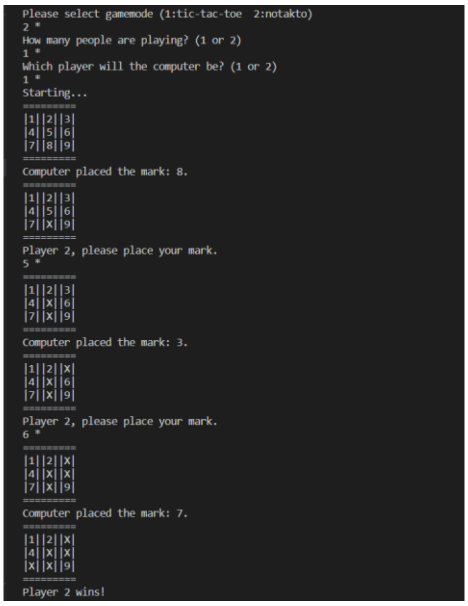


```
PS C:\projects\engg part2> .\part2.exe
Please select gamemode (1:tic-tac-toe  2:notakto)
1 *
How many people are playing?(1 or 2)
1 *
Which Player will the computer be?(Player 1 or Player 2)
2 *
Starting...
=========
|1||2||3|
|4||5||6|
|7||8||9|
=========
Player 1, please place your mark.
1 *
=========
|O||2||3|
|4||5||6|
|7||8||9|
=========
Computer placed the mark: 5.
=========
|O||2||3|
|4||X||6|
|7||8||9|
=========
Player 1, please place your mark.
7 *
=========
|O||2||3|
|4||X||6|
|O||8||9|
=========
Computer placed the mark: 4.
=========
|O||2||3|
|X||X||6|
|O||8||9|
=========
Player 1, please place your mark.
6 *
=========
|O||2||3|
|X||X||O|
|O||8||9|
=========
Computer placed the mark: 3.
=========
|O||2||X|
|X||X||O|
|O||8||9|
=========
Player 1, please place your mark.
8 *
=========
|O||2||X|
|X||X||O|
|O||O||9|
=========
Computer placed the mark: 9.
=========
|O||2||X|
|X||X||O|
|O||O||X|
=========
Player 1, please place your mark.
2 *
=========
|O||O||X|
|X||X||O|
|O||O||X|
=========
Draw game!
Again?(1:Yes, 0:No)
```



```
PS C:\projects\engg part2> .\part2.exe
Please select gamemode (1:tic-tac-toe  2:notakto)
2 *
How many people are playing? (1 or 2)
1 *
Which player will the computer be? (1 or 2)
1 *
Starting...
=========
|1||2||3|
|4||5||6|
|7||8||9|
=========
Computer placed the mark: 5.
=========
|1||2||3|
|4||X||6|
|7||8||9|
=========
Player 2, please place your mark.
2 *
=========
|1||X||3|
|4||X||6|
|7||8||9|
=========
Computer placed the mark: 7.
=========
|1||X||3|
|4||X||6|
|X||8||9|
=========
Player 2, please place your mark.
4 *
=========
|1||X||3|
|X||X||6|
|X||8||9|
=========
Computer placed the mark: 9.
=========
|1||X||3|
|X||X||6|
|X||8||X|
=========
Player 2, please place your mark.
3 *
=========
|1||X||X|
|X||X||6|
|X||8||X|
=========
Computer wins!
Again?(1:Yes, 0:No)
```

Losing AI



```
Please select gamemode (1:tic-tac-toe  2:notakto)
2 *
How many people are playing? (1 or 2)
1 *
Which player will the computer be? (1 or 2)
1 *
Starting...
=========
|1||2||3|
|4||5||6|
|7||8||9|
=========
Computer placed the mark: 8.
=========
|1||2||3|
|4||5||6|
|7||X||9|
=========
Player 2, please place your mark.
5 *
=========
|1||2||3|
|4||X||6|
|7||X||9|
=========
Computer placed the mark: 3.
=========
|1||2||X|
|4||X||6|
|7||X||9|
=========
Player 2, please place your mark.
6 *
=========
|1||2||X|
|4||X||X|
|7||X||9|
=========
Computer placed the mark: 7.
=========
|1||2||X|
|4||X||X|
|X||X||9|
=========
Player 2 wins!
```