//Vishal Singh

//Batch 3

//roll no 63

#include<bits/stdc++.h>

using namespace std;

int checking\_board(vector<int> board,int x, int o);

vector<vector<int>> possible\_move\_gen(vector<int> board,int x,int o);

int main()

{

    vector<int> board(9,0);

    int x=0,o=0;

    for(int i=0;i<9;i++)

    {

        cout<<"enter "<<i<<"th position (1/2/0):";

        int temp;

        cin>>board[i];

        if(board[i]==1)

        x++;

        else if(board[i]==2)

        o++;

    }

    int num=checking\_board(board,x,o);

    if(num==-1)

    {cout<<"invalid";return 0;}

    else cout<<num<<endl;

    vector<vector<int>> next\_moves = possible\_move\_gen(board,x,o);

    for(auto it:next\_moves)

    {

        for(auto itt:it)

        cout<<itt<<" ";

        cout<<endl;

    }

}

vector<vector<int>> possible\_move\_gen(vector<int> board,int x,int o)

{

    int turn;

    cout<<"\n Whose turn it is:";

    cin>>turn;

    int pos\_no=9-x-o;

    cout<<"possible combinations for next move: "<<pos\_no<<endl;

    vector<vector<int>>matrix(pos\_no,board) ;

    int j=0;

    for(int i=0;i<9;i++)

    {

        if(board[i]==0)

        {

            matrix[j][i]=turn;

            matrix[j].push\_back(0);

            //evaluation..

            if(i==4 )

            {

                if(( board[1]==board[7] && board[7]==turn ) || ( board[3]==board[5] && board[3]==turn )|| ( board[2]==board[6] && board[2]==turn ) || ( board[0]==board[8] && board[0]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[1]==board[7] && board[7]== abs(turn-3)) || ( board[3]==board[5] && board[3]==abs(turn-3))|| ( board[2]==board[6] && board[2]==abs(turn-3)) || ( board[0]==board[8] && board[0]==abs(turn-3)) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=4;

            }

            if(i==0)

            {

                if(( board[1]==board[2] && board[1]==turn ) || ( board[4]==board[8] && board[4]==turn )|| ( board[3]==board[6] && board[3]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[1]==board[2] && board[1]==abs(turn-3) ) || ( board[4]==board[8] && board[4]==abs(turn-3))|| ( board[3]==board[6] && board[3]==abs(turn-3) ) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=3;

            }

            if(i==2)

            {

                if(( board[0]==board[1] && board[1]==turn ) || ( board[4]==board[6] && board[4]==turn )|| ( board[5]==board[8] && board[5]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[0]==board[1] && board[1]==abs(turn-3) ) || ( board[4]==board[6] && board[4]==abs(turn-3))|| ( board[5]==board[8] && board[5]==abs(turn-3) ) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=3;

            }

            if(i==6)

            {

                if(( board[0]==board[3] && board[0]==turn ) || ( board[4]==board[2] && board[4]==turn )|| ( board[7]==board[8] && board[7]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[0]==board[3] && board[0]==abs(turn-3) ) || ( board[4]==board[2] && board[4]==abs(turn-3))|| ( board[7]==board[8] && board[7]==abs(turn-3) ) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=3;

            }

            if(i==1)

            {

                if(( board[0]==board[2] && board[0]==turn ) || ( board[4]==board[7] && board[4]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[0]==board[2] && board[0]==abs(turn-3) ) || ( board[4]==board[7] && board[4]==abs(turn-3)) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=2;

            }

            if(i==3)

            {

                if(( board[0]==board[6] && board[0]==turn ) || ( board[4]==board[5] && board[4]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[0]==board[6] && board[0]==abs(turn-3) ) || ( board[4]==board[5] && board[4]==abs(turn-3)) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=2;

            }

            if(i==5)

            {

                if(( board[8]==board[2] && board[8]==turn ) || ( board[4]==board[3] && board[4]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[8]==board[2] && board[8]==abs(turn-3) ) || ( board[4]==board[3] && board[4]==abs(turn-3)) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=2;

            }

            if(i==7)

            {

                if(( board[1]==board[4] && board[1]==turn ) || ( board[6]==board[8] && board[6]==turn ) )

                {

                    matrix[j][9]=60;

                }

                else if(( board[1]==board[4] && board[1]==abs(turn-3) ) || ( board[6]==board[8] && board[6]==abs(turn-3)) )

                {

                    matrix[j][9]=50;

                }

                else matrix[j][9]=2;

            }

        j++;}

    }

    return matrix;

}

int checking\_board(vector<int> board,int x, int o)

{

    if(abs(x-o)>1)

    return -1;

    // ternary

    int num=0;int pow3=1;

    for(int i=0;i<9;i++)

    {

        num=num+ ( board[8-i]\*pow3);

        pow3=pow3\*3;

    }

    return num;

}

Output : Non AI (ternary number calculation) + AI ( best move )

