به نام خدا

# مستندات اولین مسابقات هوشمند

# موسسه آموزش عالی سجاد

# بازی Othello


***چکیده:***

در این مقاله شرح «مسابقات هوشمند» از جمله قوانین و شرایط شرکت در آن و همچنین توضیحات بازی مربوط به این مرحله از مسابقات خواهد آمد. این مسابقات برای اولین بار و در ترم دوم سال ۱۳۸۹ در موسسه آموزش عالی سجاد برگزار می‌شود و هدف آن ارتقا، و به چالش کشیدن سطح علمی دانشجویان و ایجاد انگیزه بیشتر در آنها می‌باشد.

در هر مرحله از این مسابقات شرح قواعد یک بازی دو یا چند نفره داده می‌شود و شرکت کنندگان باید برنامه هایی بنویسند که قادر باشد انتخاب‌های صحیح و نسبتا بهینه‌ای در بازی را انجام دهد. بدیهی است برنامه ها هر چه هوشمندانه‌تر تصمیم‌گیری کنند رتبه‌ی بهتری کسب خواهند کرد.

برنامه‌ها در دسته‌های مختلفی گروه‌بندی می‌شوند (یک برنامه ممکن است در چند دسته قرار بگیرد) و به رقابت با دیگر برنامه های آن دسته خواهند پرداخت و نتایج به تفکیک هر دسته اعلام خواهد شد.

**قوانین برگزاری:**

۱- از شرکت کنندگان در گروه‌های فقط یک نفری در زمانی که متعاقبا اعلام خواهد شد ثبت نام به عمل خواهد آمد.

۲- در انتهای برگزاری مسابقات باید تیم‌های برتر، الگوریتم و کدهای خود را طی کنفرانسی توضیح دهند و همچنین مستنداتی از آن را ارائه دهند.

۳- تیم داوری، کدها و الگوریتم‌ها را به دقت بررسی کرده و در صورتی که تشخیص دهد توسط تیم شرکت کننده نوشته نشده یا هر گونه قانون دیگری نقض شده، تیم فوق از مسابقات حذف می‌شود.

٤- مسابقات بصورت حذفی و در حضور شرکت کنندگان برگزار خواهد شد و تنها در صورتی که تعداد شرکت کنندگان زیاد و برگزاری مسابقات زمانبر باشد برای سطوح ابتدایی تنها نتایج بازیها در اختیار عموم قرار خواهد گرفت.

۵- حسب صلاحدید هر یک از اساتید که برای رتبه دانشجویان خود نمره‌ای برای درس آنها در نظر گرفته اند، مسابقات در سطح گروه کلاسی آنها نیز بصورت مجزا برگزار خواهد شد.

# توضیحات بازی Othello

۱- این بازی همواره بصورت دو نفره و در صفحه‌ای مربعی و n*n که n زوج است برگزار می‌شود و سطرها و ستون‌های جدول از 0 تا n-1 شماره گذاری می شوند.

۲- در ابتدای بازی، هر بازیکن دو مهره در صفحه دارد که مهره‌های بازیکن اول مانند شکل در مختصات $(\frac{n}{2},\frac{n}{2})$ و $(\frac{n-2}{2},\frac{n-2}{2})$ و مهره های بازیکن دوم در مختصات $\left(\frac{n-2}{2},\frac{n}{2}\right)$ و $(\frac{n}{2},\frac{n-2}{n})$ قرار دارند.

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | | | | | | |
| 1 | | | | | | |
| 2 | | | 1 | 2 | | |
| 3 | | | 2 | 1 | | |
| 4 | | | | | | |
| 5 | | | | | | |

شکل ۱ – صفحه بازی با فرض n=6

۳- همواره بازیکن شماره ی ۱ شروع کننده‌ی بازی خواهد بود.

٤- بازیکنان به نوبت بازی می‌کنند و در هر نوبت سعی خواهند کرد مهره های حریف را با قرار دادن مهره‌های خود «اسیر» کنند.

۵- برای اسیر کردن یک یا چند مهره حریف که پشت سر هم و در یک خط (افقی، عمودی یا اریب) قرار دارند، باید در ابتدا و انتهای آنها مهره خودی قرار گیرد.

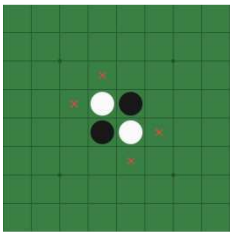
٦- مهره‌هایی که اسیر می شوند تبدیل به مهره های بازیکن اسیر کننده خواهند شد.

۷- بازیکنان زمانی مجاز به بازی هستند که بتوانند حداقل یک مهره اسیر کنند و در غیر اینصورت نوبت خود را از دست می‌دهند.

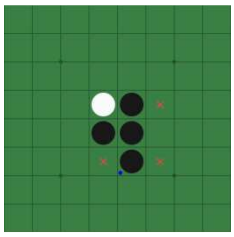
۸- بازی زمانی پایان می‌پذیرد که هیچ بازیکنی امکان بازی نداشته باشد (ممکن است قبل از پُر شدن صفحه نیز این اتفاق رخ دهد).

۹- برنده‌ی بازی کسی است که در انتها بیشترین مهره را داشته باشد.

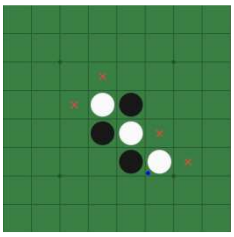
برای مثال در شکل‌های زیر ۸ حرکت دو بازیکن را مشاهده می‌کنید. در این مثال بازی را ابتدا بازیکن مشکی شروع کرده است (در مسابقه اصلی بازیکن شماره یک همیشه بازی را شروع خواهد کرد). در هر شکل علامت ضربدر نشان‌دهنده انتخاب‌های ممکن برای بازی بازیکنی است که نوبت وی می‌باشد.



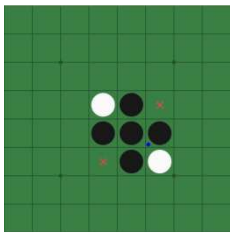
شروع بازی (در این مثال نوبت با مشکی است)



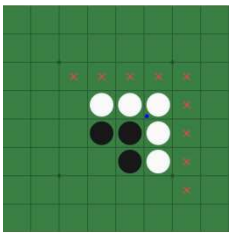
(1)



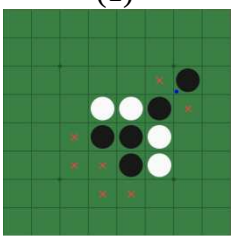
(2)



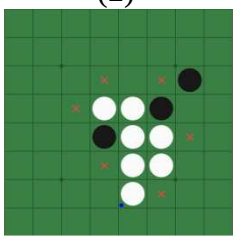
(3)



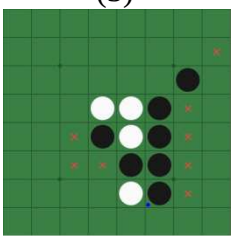
(4)



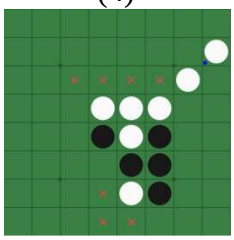
(5)



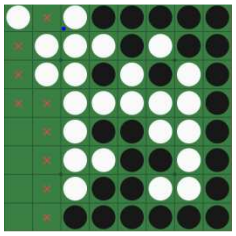
(6)



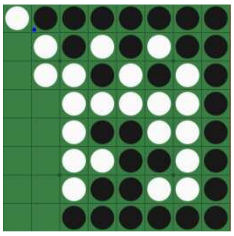
(7)



(8)



(A)



(B)

در شکل A نوبت بازی با مشکی است که بعد از بازی در شکل B مشاهده می‌شود که سفید هیچ انتخابی برای بازی ندارد و در اینصورت نوبت دوباره به بازیکن مشکی داده می‌شود.

برای اطلاعات بیشتر در رابطه با قواعد بازی به آدرس http://en.wikipedia.com/Reversi مراجعه کنید.

# قواعد برنامه‌نویسی:

در این قسمت نکاتی که در حین نوشتن برنامه لازم است رعایت شود بیان خواهد شد.

۱- برنامه‌های شما می‌تواند به هر زبانی نوشته شود و توسط هر کامپایلری کامپایل شود. برای شرکت در مسابقات باید فایل اجرایی برنامه‌ی خود را به همراه کد آن ارائه دهید که فایل اجرایی بایستی قابل اجرا در محیط ویندوز باشد و در صورت نیاز به هرگونه فایل یا پک دیگر بایستی آنها نیز تحویل گردد.

۲- برنامه باید داده‌ها را از ورودی استاندارد بخواند و در خروجی استاندارد بنویسد. برای مثال در زبان C++ به راحتی می توان از دستورات cin و cout یا printf و scanf استفاده نمود.

۳- برنامه نباید جملات اضافی و خارج از آنچه در ادامه ذکر می شود در خروجی بنویسد. مانند:

Please enter n=

٤- در ابتدا برنامه دو عدد n و P را از ورودی می‌خواند. n اندازه‌ی صفحه را مشخص می‌کند و حتما زوج و کمتر از ۲۰ خواهد بود و P شماره‌ی بازیکن است که به این برنامه اختصاص داده شده و ۱ یا ۲ خواهد بود.

۵- با توجه به اینکه همواره بازیکن ۱ شروع کننده بازی است در صورتی که P=1 باشد باید برنامه موقعیتی که می‌خواهد مهره‌ی خود را در آن قرار دهد در خروجی چاپ کند و در غیر اینصورت منتظر بازی حریف بماند. موقعیت مهره با شماره سطر و شماره ستون که با space از یکدیگر جدا شده اند، مشخص می‌شود.

٦- هر بار که برنامه‌ای بازی می‌کند، خروجی آن به ورودی برنامه‌ی حریف نیز داده می‌شود تا جریان بازی را دنبال کند.

۷- در صورتی که امکان بازی وجود نداشته باشد (هیچ مهره‌ای را نتوان اسیر کرد) باید یک جفت عدد 1- در خروجی چاپ شود. همچنین در صورتی که حریف نیز موقعیت بازی نداشته باشد یک جفت 1- می‌دهد که خروجی او به برنامه‌ی فعلی نیز داده خواهد شد.

۸- برنامه برای هر حرکت از بازی حداکثر ۲ ثانیه مهلت پاسخگویی خواهد داشت.

۹- در صورتی که هر یک از بازیکنان اشتباه بازی کنند و قواعد بازی Othello را نقض کند، بازنده بوده و حریف برنده اعلام خواهد شد.

***نکات قابل توجه:***

۱- بدیهی است موقعیت بازی و اینکه در حال حاضر هر خانه از صفحه چه وضعیتی دارد باید توسط برنامه بازیکن نگهداری و طی جریان بازی بروزرسانی شود.

۲- برنامه‌ها در حین بازی حریف suspend می‌شوند و امکان محاسبات در حین اجرای برنامه‌ی حریف را ندارند.

۳- در صورتی که برنامه در نوبت خود بیش از دو عدد در خروجی چاپ کند به عنوان ورودی حرکت بعدی‌اش در بافری ذخیره و در حرکت بعدی‌اش استفاده می‌شود.

٤- همچنین برنامه‌ی شبیه‌ساز در اختیار شرکت کنندگان قرار خواهد گرفت تا از طریق آن بتوانند برنامه‌های خود را تست و اشکال زدایی کنند.

در ادامه نمونه‌ای از برنامه‌ای که بصورت اتفاقی در یک خانه‌ی مجاز مهره قرار می‌دهد خواهد آمد. این برنامه بصورت بهینه نوشته نشده و استفاده از آن توصیه نمی‌شود.

```
#include <stdio.h>
#include <time.h>
#include <iostream>
#include <vector>
using namespace std;

#define Pair pair<int,int>
#define X first
#define Y second

int M[20][20],N;
int MyNumber, HisNumber;

int ii[]={-1,-1,-1, 0, 0, 1, 1, 1};
int jj[]={-1, 0, 1,-1, 1,-1, 0, 1};

int Straight(int i, int j, int dir, int ply, int Number)
{
    if (i < 0 || i >= N || j < 0 || j >= N) return 0;
       if (M[i][j] == Number) return 1;
    if (M[i][j] == 0) return 0;

    int t = Straight(i + ii[dir], j + jj[dir], dir, ply, Number);
    if (t > 0)
    {
        if (ply)
            M[i][j] = Number;
        return t + 1;
    }
    return 0;
}
void Play(int i, int j, int Number)
{
    for (int dir=0; dir<8; dir++)
              Straight(i + ii[dir], j + jj[dir], dir, true, Number);

    M[i][j] = Number;
}


bool CanIPlay(int i, int j)
{
    for (int dir=0; dir<8; dir++)
              if (Straight(i + ii[dir], j + jj[dir], dir, false, MyNumber) > 1)
                    return true;
    return false;
}

void MyPlay()
{
       vector<Pair > PossibleMoves;
       for(int i=0; i<N; i++)
```

هنگامی که ما یا حریف مهره ای در صفحه قرار می دهد این تابع فراخوانی می شود تا آرایه ی M که وضعیت بازی را نگهداری می کند بروزرسانی شود.

از این تابع جهت بررسی این که آیا می توان در خانه ی i,j مهره قرار داد استفاده می کنیم. البته باید چک شود که در خانه ی فوق مهره ای نباشد.

در آرایه ی PossibleMoves تمام حرکات ممکن را که در حال حاضر

```
                for(int j=0; j<N; j++)                          می توانیم بازی کنیم تولید کرده
                        if(M[i][j]==0 && CanIPlay(i,j))         تا در ادامه یکی را در خروجی چاپ
                                PossibleMoves.push_back(Pair(i,j));     کنیم.

        if(PossibleMoves.size()==0)                     در صورتی که هیچ حرکتی برای بازی کردن وجود ندارد
                cout<<"-1 -1\n";                        یک زوج اعداد 1- را در خروجی چاپ می کنیم.
        else                                            در غیر اینصورت یکی از حرکات ممکن را بصورت تصادفی انتخاب کرده
        {                                               و آن را در خروجی چاپ می کنیم. همچنین تابع Play برای بروزرسانی
                int ind=rand()%PossibleMoves.size();    موقعیت بازی فراخوانی می کنیم.
                cout<<PossibleMoves[ind].X<<" "<<PossibleMoves[ind].Y<<endl;
                Play(PossibleMoves[ind].X,PossibleMoves[ind].Y,MyNumber);
        }
}

int main()
{
        cin>>N>>MyNumber;                               ابتدا اندازه ی صفحه بازی و شماره خود را از ورودی دریافت می کنیم.

        srand(time(0));                                 برای بازی تصادفی در زمانهای مختلف از تابع srand استفاده شده است.

        memset(M,0,sizeof(M));
        M[N / 2 - 1][N / 2 - 1] = M[N / 2][N / 2] = 1;
        M[N / 2 - 1][N / 2] = M[N / 2][N / 2 - 1] = 2;

        int i,j;
        i=j=-1;

        if(MyNumber==2)
        {
                cin>>i>>j;                              در صورتی که بازیکن شماره ۲ هستیم ابتدا حریف بازی می کند و باید
                HisNumber=1;                            شماره خانه ای که در آن بازی می کند را از ورودی دریافت کنیم.
        }
        else
                HisNumber=2;

        while(1)
        {
                if(i!=-1 || j!=-1)
                        Play(i,j,HisNumber);

                MyPlay();                               فراخوانی تابع MyPlay جهت انتخاب یک خانه ی تصادفی

                cin>>i>>j;                              گرفتن حرکت بازیکن حریف از ورودی
        }
}
```