**تمرین شماره۳**

برای انجام این تمرین چیز جدیدی لازم نیست مطالعه کنید . این تمرین در واقع برای دوره ی مطالب پیشین و تست آموخته های شما در مورد برنامه نویسی C++ است.

**یک پیشنهاد** برای انجام این تمرین : سعی کنید برنامه را در چند فایل بنویسید و براش یک فایل بنویسید تا کامپایلش کنه

اطلاعات بیشتر در مورد بازی :

*http://plato.stanford.edu/entries/prisoner-dilemma/*

## IPD:

در این تمرین شما قرار است یک بازی ساده دو نفره به نام Iterated Prisoner's Dilemma یا IPD را پیاده سازی کنید.

IPD یکی از مسائل قدیمی در زمینه game theory است که تا به حال در مورد آن تحقیقات زیادی انجام شده . این بازی از حرکت های پشت سر هم دو بازیکن تشکیل شده است که هر بازیکن می تواند بین یک حرکت خود خواهانه که فقط به نفع خودش است یا حرکت غیر خودخواهانه که به نفع هر دو بازیکن است یکی را انتخاب کند.

این مدل در بسیاری از رشته ها مثل اقتصاد ، سیاست ، علوم کامپیوتر و فلسفه کاربرد های خیلی زیادی دارد.

تعریف کامل این بازی در کتاب *The Evolution of Cooperation* (1984, p.7-8) نوشته Robert Alexrod :

در بازی IPD دو بازیکن وجود دارد که هر کدام در زمان حرکت ۲ انتخاب دارد همکاری ( cooperate ) یا رد کردن ( defect) . هر کدام از این دو بازیکن باید بدون دانستن حرکت بازیکن دیگر یک انتخاب بکند . به طور کلی defect کردن به صورت شخصی بر ای هر بازیکن منفعت بیشتری دارد اما مسئله اصلی این است که اگر هر دو با هم defect کنن بدتر از این است که با هم coorporate کنند. سیستم امتیاز دهی بازی بر اساس جدول زیر است به این جدول payoff-matrixهم می گویند .

| | *player 2's move = Cooperate* | *player 2's move = Defect* |
|---|---|---|
| *player 1's move = Cooperate* | player 1's score $= 3$<br>player 2's score $= 3$ | player 1's score $= 0$<br>player 2's score $= 5$ |
| *player 1's move = Defect* | player 1's score $= 5$<br>player 2's score $= 0$ | player 1's score $= 1$<br>player 2's score $= 1$ |

اما کاری که شما باید بکنید این است که باید اسکلت برنامه ای که در اینجا می بینید کامل کنید . یعنی کامل کردن main و نوشتن ۴ تابع که prototype آنها نوشته شده است .

```
#include <iostream>
#include <cctype>
using namespace std;
// function prototypes
void update_score( char p1_move, char p2_move, int &p1_score, int &p2_score );
char all_d();
char all_c();
char tit_for_tat( char opponentsLastMove );
int main() {
        bool more = true;
        while ( more ) {
                cout << "enter C to cooperate, D to defect, Q to quit\n";
                // read user's move (step 1)
                // determine computer player's move (step 1)
                // update score for both players (step 2)
                // display both players' moves (step 1) and updated scores (step 2)
        } // end of while()
cout << "game over!\n";
} // end of main()
```

برای اینکه از دیدن این برنامه وحشت زده نشید من برنامه رو به ۵ قسمت تقسیم کردم که شما می تونید بعد از اجرای هر قسمت برنامتون رو تست کنید و اون رو توی یک فایل جدا ذخیره کنید ولی نهایتا فقط فایل کامل شده نهایی رو برای من بفرستید .

**مرحله ۱:**

- یک فایل به نام IPD.cpp درست کنید
- تابع all_d() را مطابق prototype بالا بنویسید. این تابع تنها کاری که می کند این است که کاراکتر 'D' را return می کند.
- در تابع main حرکت بازیکن را از ورودی بگیرید و در یک کاراکتر ذخیره کنید . کارکتر های مجاز اینها هستند

'C' (Cooperate),'D' (Defect) و 'Q' (quit the game).

- تابع all_d() را صدا کنید و خروجی را به عنوان حرکت کامپیوتر در یک متغیر بریزید.
- حرکت هر دو بازیکن ( بازیکن انسانی و کامپیوتر ) را چاپ کنید
- این فرایند تا وقتی که بازیکن Qوارد نکرده ادامه پیدا می کند.
- برنامه را کامپایل کنید و تست کنید و اگر مشکل داشت بر طرف کنید.

**مرحله ۲ :**

- تابع update_score() را مطابق با تعریف بالا کامل کنید این تابع به عنوان دو پارامتر اول حرکت دو بازیکن و به عنوان دو پارامتر دوم امتیاز دو بازیکن را می گیرد و به صورت ارجاع امتیاز حرکت فعلی را به امتیاز این دو بازیکن اضافه می کند.
- امتیازات بر اساس جدول Payoff_matrix محاسبه می گردد.
- این تابع را در main بعد از اینکه حرکت دو بازیکن محاسبه شد صدا کنید .
- بعد از این امتیازات هر دو بازکن را نمایش دهید .
- برنامه را کامپایل کنید و تست کنید و اگر مشکل داشت بر طرف کنید.

در این مرحله شما باید بتوانید با کامپیوتر بازی کنید . استراتژی کامپیوتر در این بازی ALL D است یعنی شما هر حرکتی که کنید کامپیوتر defect می کند . برنامه شما در اینجا باید شبیه نمونه آخر صفحه باشد .

**مرحله ۳ :**

- تابع all_c را کامل کنید ابن تابع همیشه C برمی گرداند .
- در تابع main به جای all_d ، all_c را صدا کنید .
- در واقع این تابع پیاده سازی یک استراتژی دیگر به نام ALL C هستش.
- دوباره برنامه را تست کنید . . . و در آخر امتیازات را با حالت قبلی مقایسه کنید .

**مرحله ۴:**

- تابع *tit_for_tat()* را به صورت زیر کامل کنید به عنوان حرکت کامپیوتر استفاده کنید .
- استراتژی *Tit for Tat* از حرکت *Cooperate* شروع می کند و برای حرکت های بعدی از حرکت قبل بازیکن دیگر استفاده می کند . یعنی در صورتیکه بازیکن در حرکت قبی *defect* کرده کامپیوتر در این حرکت *defect* می کند .
- با قرار دادن این تابع به جای توابع قبلی حاصل کار را مقایسه کنید .

**مرحله۵ :**

- برای بازیکن در اول برنامه یک انتخاب بگذارید که با کدام استراتژی می خواهد با کامپیوتر بازی کند .
- در مورد استراتژی های دیگر فکر کنید یا حداقل search کنید .

**نمونه اجرا شده برنامه :**

```
welcome to iterated prisonner's dilemma!
enter C to cooperate, D to defect, Q to quit
c
your move = C computer move = D
your score = 0 computer score = 5
enter C to cooperate, D to defect, Q to quit
d
your move = D computer move = D
your score = 1 computer score = 6
enter C to cooperate, D to defect, Q to quit
d
your move = D computer move = D
your score = 2 computer score = 7
enter C to cooperate, D to defect, Q to quit
q
your move = D computer move = D
your score = 2 computer score = 7
game over!
```

نکاتی در مورد نحوه انجام تمرینات :

- تمرین باید در محیط Dev C++ یا Turbo C++ نوشته شده باشد.
- تمامی فایل های مربوط به تمرین ، اعم از فایل cpp و یک فایل با نام README که حاوی توضیحات اضافی است باید در در داخل فایل فشرده قرار گرفته باشد
-
- فرزاد آقایی زاده
- www.aghaeizadeh.ir
-