تعداد از سوالات برگرفته از وب‌سایت‌های LeetCode و GeekforGeeks می‌باشند.

به سه نكته توجه كنيد:

براي هر تمريني كه در آن از شما برنامه‌اي خواسته شده است فرمت ورودي و خروجي دقيقا مشخص شده است. برنامه شما بايد دقيقا با همين فرمت كار كند تا نمره كامل بگيرد. مواردي كه با رنگ قرمز مشخص شده است توسط برنامه تست داده مي‌شود و موارد آبي رنگ را برنامه شما توليد مي‌كند. در فرمت‌هاي داده شده وقتي كه اطلاعاتي در داخل > و < ظاهر مي‌شود يعني اينكه به جاي آن يك عدد يا حرف داده خواهد شد، ولي موارد ديگر بايستي دقيقا توليد شود.

براي مثال اگر فرمت خروجي به شكل زير باشد:

Output1 = <x> : <y>

Output2 = Yes/No; <x> \* <z>

خروجی

اين خروجي‌ها، خروجي درستي است:

Output1 = 123 : 1

Output2 = No; 1000 \* 10000

خروجی

Output1 = 12 : 10000

Output2 = Yes; 100 \* 1000

خروجی

ولي اين خروجی صحیح نيست:

Output1 = 123 / 1

Output2 = YesNo; 1000 \* 10000

خروجی

سوال ۱

خروجی قطعه كد زير چيست؟ نحوه توليد خروجی را با ترسيم شكل حافظه شرح دهيد.

int j = 0, i = 0, \*p1, \*\*p2, \*\*p3;

p2 = p3 = (int \*\*)calloc(4, sizeof(int \*));

for(i = 0; i < 4; i++){

j = 2 \* i;

\*p2 = (int \*)malloc(j \* sizeof(int));

\*\*p2 = j;

p1 = \*p2;

while(j > 1){

p1++;

\*p1 = j + \*(p1 - 1);

j--;

}

p2++;

}

for(i = 3; i >= 0; i--)

for(j = 0; j < 2 \* i; j++)

printf("[%d][%d] = %d\n", i, j, \*((\*(p3 + i))+j));

سوال ۲

خروجی برنامه زیر چیست؟

#include <stdio.h>

int main()

{

int \*ptr;

int x;

ptr = &x;

\*ptr = 0;

printf(" x = %dn", x);

printf(" \*ptr = %dn", \*ptr);

\*ptr += 5;

printf(" x = %dn", x);

printf(" \*ptr = %dn", \*ptr);

(\*ptr)++;

printf(" x = %dn", x);

printf(" \*ptr = %dn", \*ptr);

return 0;

}

سوال ۳

فرض کنید برنامه زیر روی یک سیستم ۳۲ بیتی اجرا می‌شود که در آن اندازه‌ی int برابر ۴ بایت و اندازه‌ی char برابر ۱ بایت می‌باشد. خروجی برنامه را مشخص کنید.

#include <stdio.h>

int main()

{

int arri[] = {1, 2 ,3};

int \*ptri = arri;

char arrc[] = {1, 2 ,3};

char \*ptrc = arrc;

printf("sizeof arri[] = %d ", sizeof(arri));

printf("sizeof ptri = %d ", sizeof(ptri));

printf("sizeof arrc[] = %d ", sizeof(arrc));

printf("sizeof ptrc = %d ", sizeof(ptrc));

return 0;

}

سوال ۴

خروجی برنامه زیر چیست؟

#include <stdio.h>

int main()

{

float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};

float \*ptr1 = &arr[0];

float \*ptr2 = ptr1 + 3;

printf("%f ", \*ptr2);

printf("%d", ptr2 - ptr1);

return 0;

}

سوال ۵ (امتیازی)

برنامه‌ای بنویسید که با دریافت یک مجموعه از اعداد تمام زیر مجموعه‌های آن را چاپ کند.

3

1 2 3

ورودی

{ }

{ 1 }

{ 2 }

{ 3 }

{ 1 2 }

{ 2 3 }

{ 1 3 }

{ 1 2 3}

خروجی

سوال ۶

آرایه‌ای از اعداد صحیح به شما داده می‌شود. برنامه‌ای بنویسید که کوچکترین عدد صحیح مثبت که در این آرایه نیامده است را پیدا کند. در نظر داشته باشید که این آرایه می‌تواند درایه‌های منفی یا صفر هم داشته باشد.

3

1 2 0

ورودی

3

خروجی

4

3 4 -1 1

ورودی

2

خروجی

5

7 8 9 11 12

ورودی

1

خروجی

سوال ۷

برنامه‌ای بنویسید دو رشته را از کاربر گرفته و مشخص کند آیا رشته دوم در رشته اول وجود دارد یا خیر و در صورت وجود آن را حذف کرده و رشته جدید را چاپ کند. وجود داشتن به این معنا می‌باشد که رشته دوم دقیقا و بدون هیچ کاراکتر اضافه‌ای در رشته اول وجود داشته باشد. در نظر داشته باشید که رشته‌ها می‌توانند شامل فاصله باشند اما حداکثر ۲۰۰ کاراکتر داشته و در یک خط می‌باشند. در نظر داشته باشید که در صورتی که این رشته وجود نداشت رشته‌ی اولیه را چاپ کند.

Hello World

Hello

ورودی

World

خروجی

Hello World

l World

ورودی

Hello World

خروجی

Hello World

o World

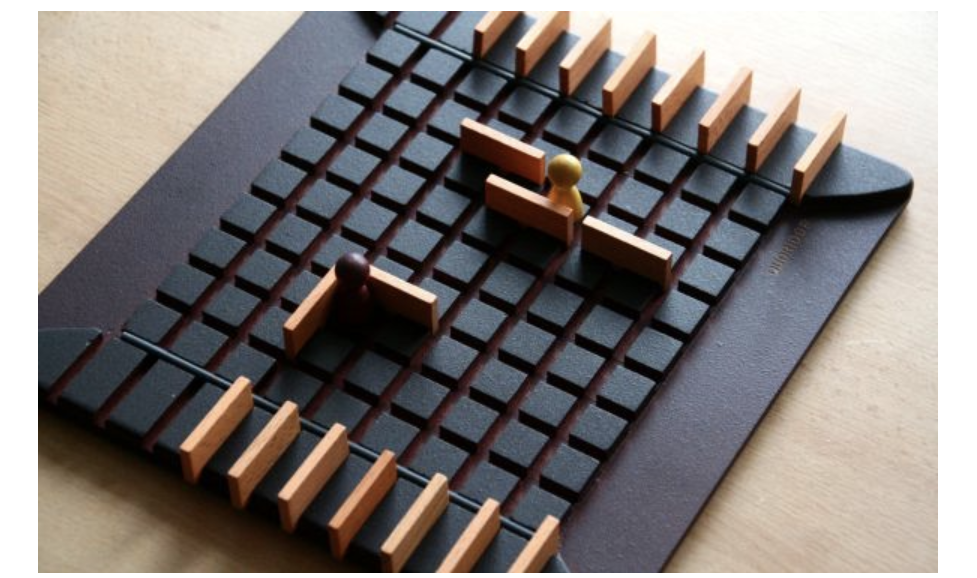
ورودی

Hell

خروجی

سوال ۸ (امتیازی)

بازی کوریدور



قوانین بازی:

قوانین بازی به طور خلاصه به شرح زیر است:

۱. صفحه بازی یک صفحه ۹×۹ است و هر بازیکن یک مهره دارد که در وسط سمت خود قرار می گیرد.

۲. هدف بازی این است که هر یک از مهره ها به یکی از خانه های سمت مقابل خود برسد.

۳. هر دیوار مسیر ۲ خانه را در یک جهت می بندد.

۴. مهره‌ها نمی‌توانند از روی دیوار بپرند.

۵. دیوارها نمی‌توانند یکدیگر را قطع کنند.

۶. دیوارها پس از قرار داده شدن قابل حذف نیستند.

۷. تعداد کل دیوارهای بازی ۲۰ عدد است که به طور مساوی میان بازیکنان پخش می‌گردد.

۸. هر بازیکن در هر حرکت میتواند مهره خود را در یکی از ۴ جهت اصلی به اندازه یک واحد حرکت داده و یا یک دیوار در صفحه قرار دهد.

۹. در سطر ها و ستون های کناری بازی دیوار نمی‌تواند قرار گیرد.

برای آشنایی بیشتر می‌توانید از طریق لینک زیر بازی را انجام دهید.

<https://danielborowski.github.io/site/quoridor-ai/display.html>

همچنین در لینک زیر توضیحات بیشتری در خصوص بازی آمده است.

<https://en.wikipedia.org/wiki/Quoridor>

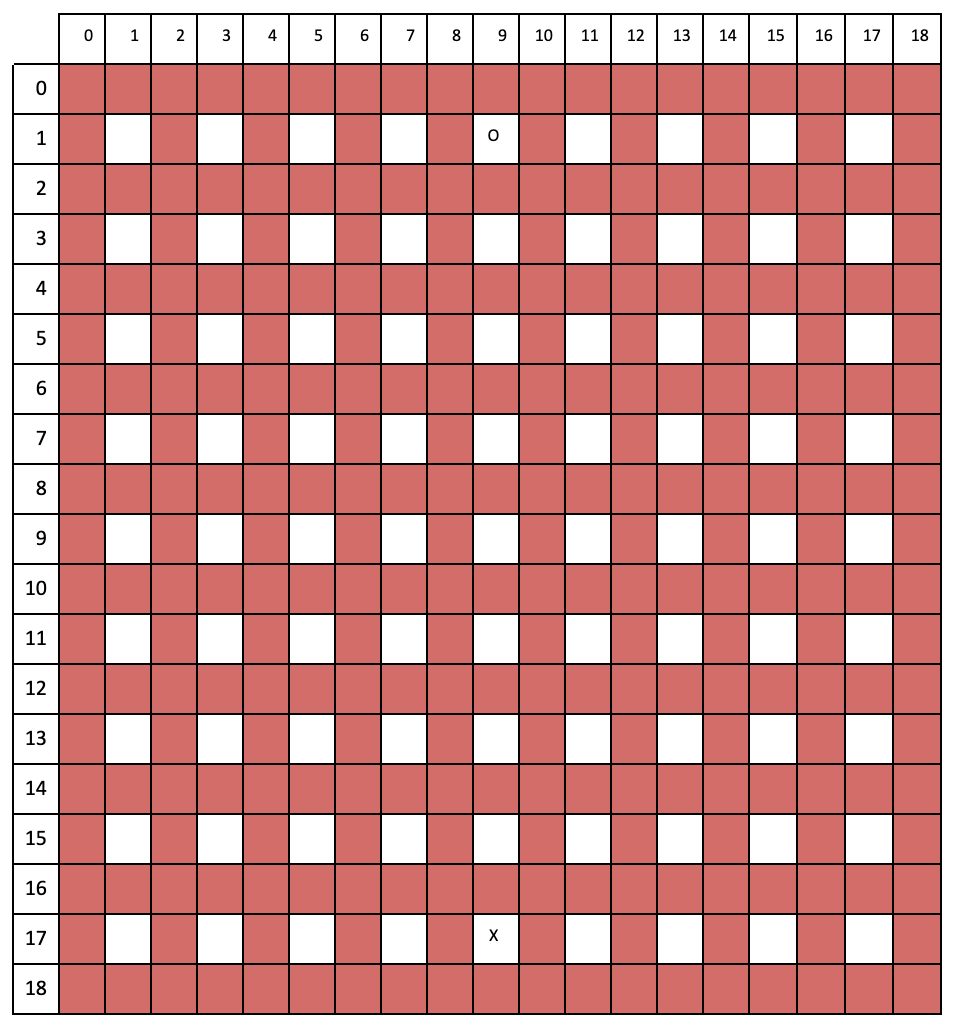
در این تمرین شما قرار است بازی کوریدور را با توجه به قوانین گفته شده در بالا پیاده سازی کنید. ابتدا تعدادی توابع که در ادامه مشخصات آن آمده است را پیاده سازی کنید و سپس با استفاده از این توابع بازی را به طور کامل پیاده سازی کنید. برای راحتی نحوه نگه‌داری و چاپ جدول بازی پیاده سازی شده و در پیوست آمده است، و شما باید با استفاده از توابع و متغیر‌های تعریف شده سایر توبع را پیاده سازی کنید.

توضیحات نقشه بازی:

نقشه بازی در ابتدا خالی بوده و صرفا همان صفحه ۹ در ۹ بازی است.

احتمالا برای ذخیره وضعیت خانه های نقشه بازی، نیاز به آرایه ۹ در ۹ داریم.

اما با این تعداد تنها میتوان وضعیت خانه ها را ذخیره کرد. برای اینکه بتوانیم دیوارها را نیز در نقشه‌مان داشته باشیم، نیاز به خانه های بیشتری داریم. به تصویر زیر دقت کنید:



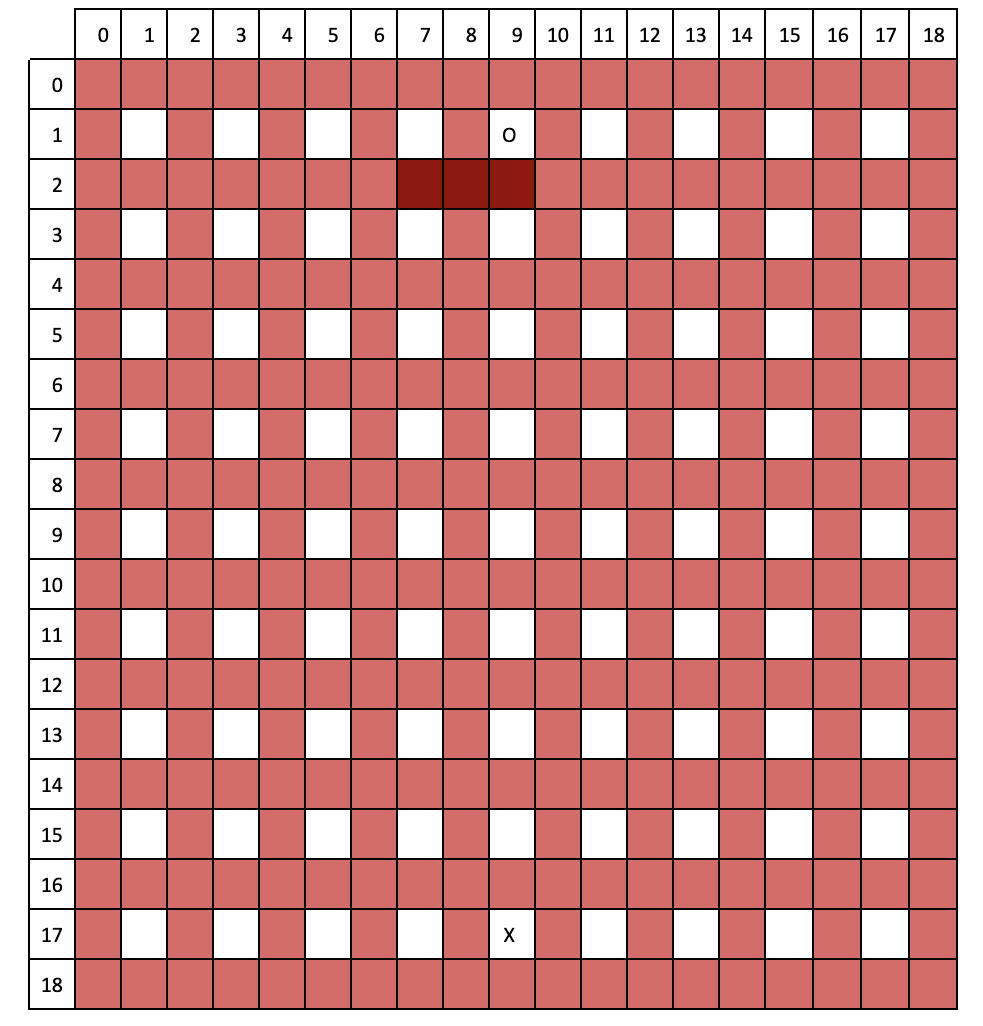
بخش های سفید جدول همان نقاطی هستند که مهره ها(جناب X و جناب O) قرار میگیرند. و باقی بخش‌ها محل قرار گیری دیوار‌ها هستند.

بیاید فرض کنیم جناب X می‌خواهد به عنوان اولین حرکت بازی، راه جناب O رو ببندد. احتمالا یک دیوار افقی باید جلوش بگذارد.

کلا به چه ترتیبی یک دیوار رو می‌گذاریم در نقشه؟ روند به این شکل است که هر جایی که قرار است دیوار باشد را در نظر می‌گیریم،‌ جایی قسمت بالا و چپ دیوار قرار می‌گیرد را به عنوان محور دیوار در نظر میگیریم. از روی جایی که نقطه قرار دارد می‌شود فهمید که دیوار عمودی است یا افقی

برای مثال جناب X می‌تواند توی نقطه x:2 و y:7 یک دیوار افقی بگذارد.

بعد از قرار داده شدن دیوار، نقشه بازی به این شکل درمیاد:



توضیحات توابع و متغیرها:

#define SIZE 9:

تعیین می‌کند که ابعاد صفحه بازی چند در چند باشد. در حالت مورد نظر ما بازی ۹×۹ است.

#define ROW\_BLOCK\_SIZE 4:

تعیین میکند که هر خانه در حالت نمایش چند خط ارتفاع داشته باشد.

#define COL\_BLOCK\_SIZE 8:

تعیین میکند که هر خانه در حالت نمایش چند کاراکتر عرض داشته باشد.

int map[2\*SIZE+1][2\*SIZE+1]:

نقشه بازی که در آن محل دیوار ها و مهره ها مشخص میشود.

int players[2][2]:

متغیری که در هر سطر آن یک بازیکن نگه داری میشود و در هر سطر آن x,y محل قرار گیری مهره مشخص شده است.

char players\_pawn[2] = {'x', 'o'}:

متغیری که تعیین میکند هر مهره در صفحه بازی با چه علامتی نشان داده شود.

char visual\_map[4\*SIZE+1][8\*SIZE+1]:

آرایه ای که طبق آن صفحه بازی نمایش داده میشود.

void print():

این تابع از روی visual\_map، صفحه بازی را روی کنسول چاپ میکند.

void init\_table():

این تابع، visual\_map را مقداردهی میکند.

void add\_block\_at(int i,int j):

این تابع در i و j ورودی داده شده مربوط به map، دیوار قرار میدهد. همچنین باید visual\_map را آپدیت کند.

void add\_pawn(int player,int i, int j):

این تابع شماره مهره را ورودی میگیرد(با توجه به آرایه players)، و در i و j ورودی داده شده مربوط به map، مهره را قرار میدهد. همچنین باید visual\_map را نیز اپدیت کند(اینکه چه علامتی در آن نقطه قرار دهد توسط آرایه players\_pawn مشخص میشود)

void remove\_pawn(int i, int j):

این تابع i و j مربوط به map را به عنوان ورودی میگیرد و مهره را از آن نقطه حذف میکند. همچنین باید visual\_map را نیز اپدیت کند.

bool check\_move(int player, int direction):

این تابع شماره مهره را ورودی میگیرد(با توجه به آرایه players)، و باید چک کند که طبق قوانین بازی، آیا حرکت در جهت ورودی داده شده مجاز است یا خیر.

جهت ها {0,1,2,3} هستند که از چپ به راست نشان دهنده بالا، راست، پایین و چپ هستند.

void apply\_move(int player, int direction):

این تابع شماره مهره را ورودی میگیرد(با توجه به آرایه players)، و باید آن مهره را در جهت ورودی داده شده حرکت دهد. یعنی مکان مهره را در players اپدیت کند. و map و visual\_map را طبق این حرکت اپدیت کند.

توجه کنید که در این تابع فرض بر این است که حرکت مجاز است. یعنی نیاز به فراخوانی تابع check\_move برای چک کردن اینکه حرکت مجاز است یا خیر، نیست.

bool check\_block(int place[], int direction):

این تابع یک آرایه ۲تایی مکان(با توجه به آرایه map) و همچنین یک جهت را ورودی میگیرد، و باید چک کند که طبق قوانین بازی، آیا قرار دادن دیوار با این جهت در این مکان، مجاز است یا خیر.

جهت ها {0,1,2,3} هستند که از چپ به راست نشان دهنده بالا، راست، پایین و چپ هستند.

void apply\_block(int place[], int direction):

این تابع یک آرایه ۲تایی مکان(با توجه به آرایه map) و همچنین یک جهت را ورودی میگیرد، و باید دیوار را در آن نقطه و با آن جهت در map و visual\_map قرار دهد.

bool check\_inside(int i, int j):

یک i و j مربوط به map را ورودی میگیرد و باید چک کند که آیا این نقطه داخل صفحه بازی قرار میگیرد یا خیر.

bool end\_game():

این تابع چک میکند که بازی به اتمام رسیده یا خیر و یک True/False خروجی میدهد.

int main():

در این تابع نیز مدیریت بازی انجام میشود. یعنی نوبت دهی به کاربر ها، ورودی گرفتن حرکت مورد نظر کاربر، چک کردن اینکه ورودی کاربر معتبر باشد و انجام حرکت مورد نظر کاربر نیز در این تابع قرار دارد.

همچنین تابع چک کردن پایان بازی باید توسط شما پیاده سازی شود.