مدرس: رامتین خسروی طراح: رامتین خسروی

موعد تحویل: ۱۴ مهر ۱۳۹۳

قسمت اول: بازی X-O

بازی تیک‌تک‌تو[1] یا آنچه ما بیشتر آن را به نام ایکس‌او می‌شناسیم به این ترتیب است که دو بازیکن هر یک با یکی از نمادهای X یا O سعی می‌کنند یک سطر، یک ستون یا یک قطر یک جدول ۳×۳ را با نماد خود پر کنند. دو بازیکن به ترتیب بازی می‌کنند و در هر نوبت می‌توانند نماد خود را در یک خانه‌ی خالی از جدول قرار دهند. بازیکن X شروع‌کننده‌ی بازی است و تا زمانی که یکی از دو بازیکن برنده شوند یا کل جدول پر شود بازی ادامه می‌یابد.

هدف این قسمت، نوشتن این بازی است طوری که از توابع برای شکست بالا به پایین برنامه استفاده شود. در این قسمت ورودی و خروجی به شکل متنی است و در قسمت بعدی به آن گرافیک اضافه خواهد شد.

الگوریتم کلی این بازی به این شکل است:

خانه‌های جدول را به خالی مقداردهی کن
نماد فعلی را به ایکس مقداردهی کن
تا موقعی که بازی تمام نشده تکرار کن
  یک مختصات خانه‌ی مورد نظر بازیکن فعلی را از ورودی بگیر (تکرار در صورت غیرمجاز بودن)
  خانه‌ی جدول را با نماد فعلی علامت‌گذاری کن
  اگر بازیکن فعلی برنده است بازی را تمام کن
  نماد فعلی را عوض کن

این بازی را با رعایت نکات زیر پیاده‌سازی کنید:

- تا حد امکان عملیات را در قالب توابع مستقل انجام دهید. مثلاً بند اول (خالی کردن جدول) را در یک تابع مستقل پیاده‌سازی کنید. توابع شما ممکن است خودشان از توابع کوچک‌تری استفاده کنند.
- به هیچ‌وجه از متغیرهای سراسری[2] استفاده نکنید. هر تابع اطلاعات مورد نیازش را از پارامترهایش دریافت و خروجی‌های خود را از طریق مقدار بازگشتی یا پارامترهای خود به بیرون برگرداند.
- در نام‌گذاری خوانا برای توابع و پارامترهای دقت کنید.
- پیش از هر نوبت بازی وضعیت صفحه را نمایش دهید و مختصات خانه‌ها را به شکل سطر و ستون دریافت کنید (خانه‌ی بالا سمت چپ مختصات ۱ و ۱ دارد). به عنوان نمونه ابتدای اجرای برنامه‌ی شما ممکن است به این شکل باشد:

[1] tic-tac-toe
[2] global

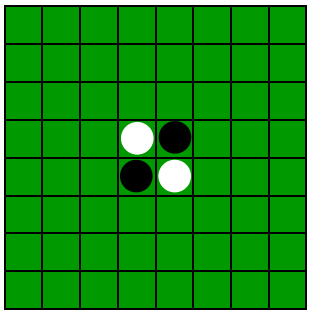
```
 | |
-+-+-
 | |
-+-+-
 | |
Player X's turn: 2 2
 | |
-+-+-
 |X|
-+-+-
 | |
Player O's turn: 2 2
Invalid move!
Player O's turn: 1 3
 | |O
-+-+-
 |X|
-+-+-
 | |
```

قسمت دوم: اضافه کردن واسط کاربری گرافیکی

در این قسمت از کتابخانه‌ی RSDL استفاده می‌شود[۳]. با استفاده از این کتابخانه، یک واسط کاربری گرافیکی برای بازی خود ایجاد کنید. برای این کار می‌توانید از مثال‌های قرار گرفته روی سایت درس استفاده کنید. بازیکن‌ها باید با کلیک کردن روی خانه‌های جدول خانه‌ی مورد نظر خود را انتخاب نمایند. اگر خانه‌ی انتخاب شده غیرمجاز بود کاری انجام ندهید. هنگامه خاتمه‌ی بازی از طریق پیغامی در کنسول نتیجه‌ی بازی را اعلام نمایید.

قسمت سوم: بازی اتلّو

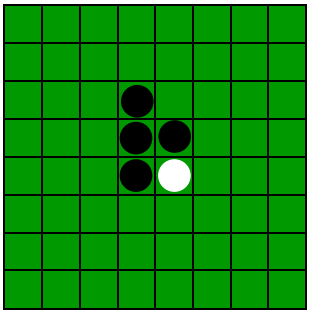
در این قسمت به شکل گرافیکی بازی اتلّو[۴] را پیاده‌سازی کنید. شرح دقیق این بازی در صفحه ویکی‌پدیای این بازی قابل دسترسی است. به طور خلاصه، این بازی توسط دو بازیکن روی یک صفحه‌ی ۸×۸ انجام می‌شود. یکی از بازیکن‌ها مهره‌های سفید و دیگری مهره‌های سیاه در اختیار دارد. وضعیت اولیه‌ی صفحه به این شکل است:



[۳] برای یادگیری کار با این کتابخانه به صفحه درس برنامه‌سازی پیشرفته در CECM مراجعه نمایید.

[4] Othello

بازی را بازیکن با مهره‌های تیره شروع می‌کند. در هر نوبت، بازیکن مجاز است مهره‌ی خود را در خانه‌ای بگذارد که تعدادی مهره‌ی حریف در یک خط مستقیم (سطر، ستون، یا قطر) بین دو مهره‌ی او قرار بگیرد. در نتیجه‌ی این بازی، مهره‌های مذکور به رنگ مهره‌های بازیکن تغییر رنگ می‌دهند. به عنوان مثال اگر اولین حرکت سیاه در خانه‌ی سطر سوم و ستون چهارم باشد وضعیت صفحه به این شکل خواهد شد:



توجه داشته باشید که ممکن است در یک حرکت مهره‌های حریف در بیش از یک خط مستقیم بین دو مهره‌ی بازیکن قرار بگیرند که تمام آنها تغییر رنگ خواهند داد. اگر بازیکنی طبق قانون بالا امکان گذاشتن مهره‌ی خود را نداشته باشد، نوبت خود را از دست می‌دهد. بازی هنگامی خاتمه می‌یابد که یا تمام خانه‌های صفحه پر باشد یا هیچ بازیکنی امکان حرکت نداشته باشد. در پایان بازی بازیکنی که بیشترین تعداد مهره را داشته باشد برنده است.

این بازی را با در نظر گرفتن نکاتی که در قسمت‌های اول و دوم تمرین عنوان شدند با استفاده از RSDL بنویسید.

نحوه‌ی تحویل

فایل قسمت i ام را به نام A2-SID-i.cpp ذخیره کنید. سه فایل مربوط به سه قسمت را در یک پوشه به نام A2-SID قرار داده و آن را با فرمت zip آرشیو کنید و در نهایت فایلی با نام A2-SID.zip را در سایت درس آپلود کنید. (SID پنج رقم آخر شماره‌ی دانشجویی شماست. به عنوان مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۲۱۲۳ است، نام فایل شما باید A2-92123.zip باشد.)

بارم‌بندی

قسمت اول: ۱۰ نمره درستی عملکرد — ۲۰ نمره خوانایی و ساختار مناسب برنامه
قسمت دوم: ۱۵ نمره درستی عملکرد — ۱۵ نمره خوانایی و ساختار مناسب برنامه
قسمت سوم:۲۰ نمره درستی عملکرد — ۲۰ نمره خوانایی و ساختار مناسب برنامه

دقت کنید

- برنامه‌ی شما باید در سیستم‌عامل لینوکس نوشته شده، با مترجم g++ ترجمه شود.
- به فرمت و نام فایل‌های خود دقت کنید. در صورتی که هر یک از موارد گفته شده رعایت نشود، نمره‌ی صفر برای شما در نظر گرفته می‌شود.
- در صورت کشف تقلب در کل و یا قسمتی از تمرین، برای هر دو طرف نمره‌ی ۱۰۰ ـ منظور خواهد شد.