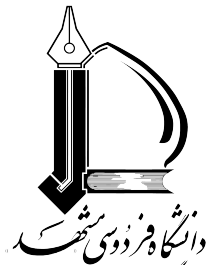


دانشگاه فردوسی مشهد

دانشکده مهندسی

پروژه پایان‌ترم درس مبانی کامپیوتر و برنامه‌سازی

استاد:

دکتر مصطفی نوری بایگی

اعضای حل تمرین:

محسن غلامی

زهرا علیپور

هلیا بهشتی

نگار بهرامپور

متین نظام‌الملک

هادی شرقی

رضا منصوری

ابوالفضل عمارلو

سجاد زیرک

آرمان بیجاری

نیمسال:

پاییز ۱۴۰۴

مهلت نهایی تحویل پروژه:

۱۲ بهمن ماه – **غیرقابل تمدید**

فهرست مطالب

۱ مقدمه و معرفی پروژه ۲

۲ بخش اول: ایجاد نقشه (۱۰٪) ۳
۱.۲ نمونه ورودی ۳
۲.۲ نمونه خروجی ۴

۳ بخش دوم: منطق بازی دو نفره (۳۰٪) ۴

۴ بخش سوم: بازی با کامپیوتر (۱۵٪) ۵

۵ بخش چهارم: ذخیره و بازی مجدد (۱۵٪) ۵

۶ بخش پنجم: طلسم‌ها و پاداش‌ها (۳۰٪) ۶
۱.۶ طلسم‌ها ۶
۲.۶ هدیه‌ها ۶

۷ موارد نمره اضافه ۷

۸ نکات ۸

۹ معیارهای نمره‌دهی پروژه ۹
۱.۹ فاز اول: ایجاد نقشه و نمایش بورد (10 نمره) ۹
۲.۹ فاز دوم: منطق بازی دو نفره (30 نمره) ۹
۳.۹ فاز سوم: بازی با کامپیوتر (15 نمره) ۹
۴.۹ فاز چهارم: ذخیره و بازی مجدد (15 نمره) ۹
۵.۹ فاز پنجم: طلسم‌ها و پاداش‌ها (30 نمره) ۱۰
۶.۹ ضریب تمیزی کد و تسلط ۱۰

۱۰ نکات تکمیلی ۱۰

۱ مقدمه و معرفی پروژه

در این پروژه، هدف پیاده‌سازی بازی رومیزی Quoridor با استفاده از زبان برنامه‌نویسی C است. بازی Quoridor یک بازی فکری نوبتی برای ۲ تا ۴ بازیکن می‌باشد که تمرکز اصلی آن بر برنامه‌ریزی مسیر، تصمیم‌گیری استراتژیک و محدودسازی حرکت حریف است.

در این بازی، هر بازیکن یک مهره و تعداد مشخصی دیوار در اختیار دارد. هدف اصلی هر بازیکن این است که مهره‌ی خود را به ضلع مقابل صفحه‌ی بازی برساند، در حالی که هم‌زمان با استفاده از دیوارها، مسیر حرکت بازیکن حریف را طولانی‌تر یا محدودتر می‌کند. برنده‌ی بازی بازیکنی است که زودتر از سایرین به ضلع هدف خود برسد.

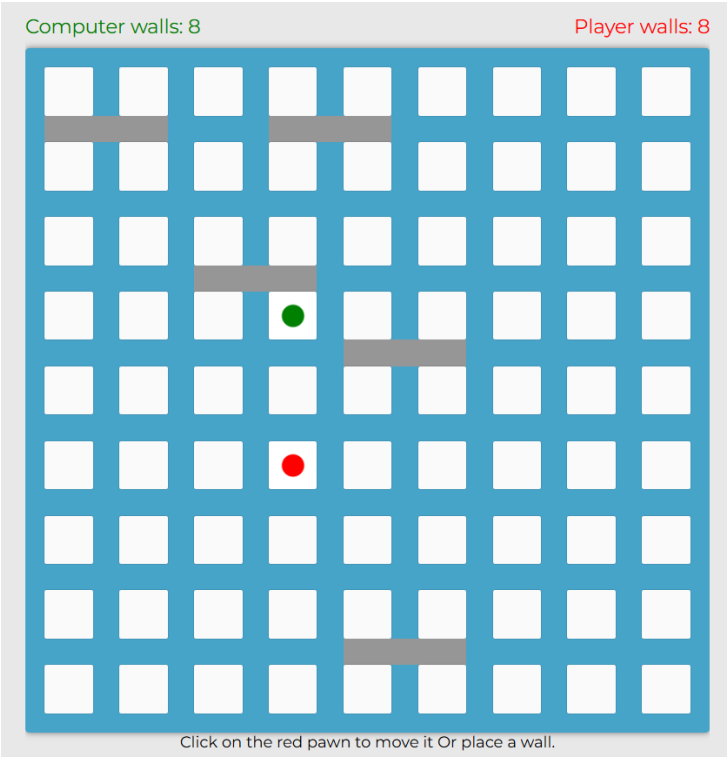
در هر نوبت، بازیکن می‌تواند یکی از دو عمل زیر را انجام دهد:

- حرکت دادن مهره‌ی خود به یکی از خانه‌های مجاور به‌صورت افقی یا عمودی
- قرار دادن یک دیوار در محل مجاز بین خانه‌های صفحه‌ی بازی

هر دیوار طولی برابر با دو خانه دارد و باید به‌صورت افقی یا عمودی بین خانه‌ها قرار گیرد. تعداد دیوارهای هر بازیکن محدود است و هنگام قرار دادن دیوارها باید دقت شود که همواره حداقل یک مسیر معتبر برای رسیدن هر بازیکن به ضلع هدف وجود داشته باشد؛ در غیر این صورت، قرار دادن دیوار در آن موقعیت مجاز نخواهد بود.

برای آشنایی بیشتر با قوانین بازی و تجربه‌ی نسخه‌ی آنلاین آن، می‌توان از لینک زیر استفاده کرد:

http://quoridor.di.uoa.gr/



شکل ۱: نمونه‌ای از صفحه‌ی بازی Quoridor و نحوه‌ی قرارگیری مهره‌ها و دیوارها

۲ بخش اول: ایجاد نقشه (۱۰٪)

در این بخش، هدف نمایش نقشه‌ی بازی به‌همراه جایگاه مهره‌ها و دیوارها است. نقشه‌ی بازی باید با استفاده از کاراکترهای ASCII چاپ شود و تلاش شود بهترین و خواناترین نمایش ممکن برای صفحه‌ی بازی انتخاب گردد.

ورودی این بخش از طریق یک فایل متنی دریافت می‌شود. در این فایل، اطلاعات زیر به‌ترتیب مشخص‌شده قرار دارند:

- ابعاد نقشه‌ی بازی
- موقعیت بازیکن اول و بازیکن دوم
- تعداد دیوارهای هر بازیکن
- مختصات هر دیوار و نوع قرارگیری آن (افقی یا عمودی)

هر دیوار طولی برابر با دو خانه دارد و باید بین خانه‌های نقشه قرار گیرد. پس از خواندن اطلاعات از فایل متنی، برنامه موظف است نقشه‌ی بازی را در قالب یک بورد گیم متنی نمایش دهد، به‌طوری که موقعیت بازیکنان و دیوارها به‌وضوح قابل تشخیص باشد. برای طراحی مناسب ساختار داده‌های مربوط به نقشه‌ی بازی، استفاده از struct و enum مطابق توضیحات ارائه‌شده در بخش ن‌ک‌ات (نکته‌ی ۸) توصیه می‌شود.

۱.۲ نمونه ورودی

برای مثال، محتوای فایل ورودی می‌تواند به‌صورت زیر باشد:

```
10
0 1
7 3
1
1 1 H
1
5 5 V
```

در این ورودی:

- اندازه‌ی نقشه برابر با ۱۰ در ۱۰ است.
- بازیکن شماره‌ی یک در سطر ۰ و ستون ۱ قرار دارد.
- بازیکن شماره‌ی دو در سطر ۷ و ستون ۳ قرار گرفته است.

- بازیکن شماره‌ی یک یک دیوار افقی دارد که از مختصات (1,1) شروع می‌شود.
- بازیکن شماره‌ی دو یک دیوار عمودی در مختصات (5,5) قرار داده است.

در صورت وجود داده‌ی نامعتبر در فایل ورودی، برنامه باید از اعمال آن جلوگیری کرده و پیام خطای مناسب نمایش دهد، بدون آن‌که اجرای برنامه متوقف شود.

۲.۲ نمونه خروجی

در خروجی، برنامه باید یک بورد ۱۰ در ۱۰ را نمایش دهد که در آن:

- جایگاه هر دو بازیکن مشخص باشد.
- دیوارها مطابق با مختصات و جهت داده‌شده رسم شده باشند.
- طول هر دیوار دقیقاً دو خانه در نظر گرفته شود.

۳ بخش دوم: منطق بازی دو نفره (۳۰٪)

در این بخش، هدف پیاده‌سازی منطق کامل بازی دو نفره است. بازی به‌صورت نوبتی انجام می‌شود و در هر نوبت، بازیکن جاری باید تصمیم بگیرد که مهره‌ی خود را حرکت دهد یا در صورت داشتن دیوار، یک دیوار در محل مجاز قرار دهد. نوبت‌ها به‌ترتیب بین بازیکن شماره‌ی یک و بازیکن شماره‌ی دو تا پایان بازی ادامه می‌یابد.

در ابتدای بازی، بازیکن شماره‌ی یک در پایین‌ترین سطر و در ستون وسط صفحه‌ی بازی قرار می‌گیرد و بازیکن شماره‌ی دو در بالاترین سطر، دقیقاً در مقابل بازیکن شماره‌ی یک، مستقر می‌شود.

در ورودی این بخش، اطلاعات زیر از کاربر دریافت می‌شود:

- ابعاد تخته‌ی بازی
- تعداد دیوارهای مجاز برای هر بازیکن
- نام بازیکن شماره‌ی یک و بازیکن شماره‌ی دو

پس از مقداردهی اولیه، بازی با نوبت بازیکن شماره‌ی یک آغاز می‌شود. در هر دور، بازیکن باید یکی از دو عمل زیر را انتخاب کند:

- حرکت دادن مهره‌ی خود به یک خانه‌ی مجاور مجاز
- قرار دادن یک دیوار، در صورتی که دیوار در اختیار داشته باشد

حرکت مهره به‌صورت افقی یا عمودی و تنها به خانه‌های مجاور مجاز است. در صورتی که مهره‌ی دو بازیکن در خانه‌های مجاور و روبه‌روی یکدیگر قرار گیرند، بازیکن جاری موظف است در صورت امکان از روی مهره‌ی حریف پرش کرده و به خانه‌ی پشت آن منتقل شود.

در حالتی که پرش مستقیم به دلیل وجود دیوار در پشت مهره‌ی حریف امکان‌پذیر نباشد، بازیکن مجاز است مهره‌ی خود را به یکی از خانه‌های مورب مجاور (بالا-چپ یا بالا-راست، و یا پایین-چپ یا پایین-راست متناسب با جهت حرکت) منتقل نماید، مشروط بر آن‌که آن خانه توسط دیوار مسدود نشده باشد.

در سایر شرایط، انجام حرکت مورب یا پرش غیرمجاز تلقی شده و برنامه باید از انجام آن جلوگیری نماید.

در صورتی که تعداد دیوارهای یک بازیکن به صفر برسد، برنامه باید از قرار دادن دیوار توسط آن بازیکن جلوگیری کرده و پیام خطای مناسب نمایش دهد.

همان‌طور که در بخش قوانین بیان شد، دیوارها تنها در مکان‌هایی قابل قرار دادن هستند که برای هر دو بازیکن حداقل یک مسیر معتبر جهت رسیدن به ضلع هدف وجود داشته باشد. در صورتی که قرار دادن یک دیوار باعث مسدود شدن کامل مسیر هر یک از بازیکنان شود، آن دیوار نامعتبر تلقی شده و نباید اجازه‌ی قرارگیری داده شود.

برای تشخیص معتبر بودن مسیرها، می‌توان از الگوریتم‌های مسیریابی در گراف مانند DFS و BFS استفاده کرد. منظور از وجود مسیر معتبر، وجود حداقل یک مسیر پیوسته از موقعیت فعلی مهره تا هر یک از خانه‌های ضلع هدف بازیکن، بدون عبور از دیوارها است. منظور از ضلع هدف، هر یک از خانه‌های موجود در ردیف نهایی مقابل موقعیت شروع بازیکن می‌باشد و رسیدن مهره به هر یک از این خانه‌ها به‌منزله‌ی پایان بازی و پیروزی آن بازیکن است.

۴ بخش سوم: بازی با کامپیوتر (۱۵٪)

در این بخش، امکان بازی کاربر با کامپیوتر فراهم می‌شود. کامپیوتر در هر نوبت خود باید با استفاده از توابع تولید مقادیر تصادفی random تصمیم بگیرد که مهره‌ی خود را حرکت دهد یا یک دیوار در محل مجاز قرار دهد.

با وجود تصادفی بودن تصمیم‌گیری کامپیوتر، تمامی قوانین بازی باید به‌طور کامل رعایت شوند. به‌عبارت دیگر، کامپیوتر تنها مجاز به انجام حرکت‌هایی است که مطابق با قوانین حرکت مهره و قواعد قرار دادن دیوار باشند. در صورتی که یک تصمیم تصادفی منجر به حرکت یا قرار دادن دیوار نامعتبر شود، آن تصمیم باید رد شده و تصمیم جدیدی انتخاب گردد.

در ابتدای اجرای بازی، برنامه باید از کاربر سوال کند که آیا تمایل دارد بازی را در حالت دو نفره انجام دهد یا با کامپیوتر بازی کند. در صورت انتخاب حالت بازی با کامپیوتر، یک بازیکن توسط کاربر کنترل شده و بازیکن دیگر توسط کامپیوتر هدایت می‌شود.

۵ بخش چهارم: ذخیره و بازی مجدد (۱۵٪)

در این بخش، برنامه باید قابلیت ذخیره‌سازی و بازیابی وضعیت بازی را فراهم کند. بازیکنان می‌توانند در هر مرحله از بازی، در صورت تمایل، وضعیت فعلی بازی را در یک فایل باینری ذخیره نمایند.

در ابتدای اجرای برنامه، در صورتی که فایل ذخیره‌شده‌ای وجود داشته باشد، کاربر باید این امکان را

داشته باشد که یکی از بازی‌های ذخیره‌شده را انتخاب کرده و بازی را دقیقاً از همان وضعیت ادامه دهد. در صورت عدم انتخاب بازی ذخیره‌شده، بازی جدید آغاز می‌شود.

فرآیند ذخیره‌سازی و بازیابی باید **به‌صورت کامل در قالب فایل باینری** انجام شود و استفاده از فایل‌های متنی برای این بخش مجاز نمی‌باشد.

اطلاعات ذخیره‌شده باید شامل تمامی داده‌های لازم برای ادامه‌ی صحیح بازی باشد، از جمله موقعیت مهره‌ها، وضعیت دیوارها، تعداد دیوارهای باقی‌مانده‌ی هر بازیکن و نوبت بازیکن جاری. فرمت دقیق فایل باینری آزاد بوده، اما باید به‌گونه‌ای طراحی شود که امکان ذخیره و بازیابی صحیح داده‌ها در اجرای‌های بعدی برنامه وجود داشته باشد.

۶ بخش پنجم: طلسم‌ها و پاداش‌ها (۳۰٪)

در این بخش، در هر نوبت از بازی، به‌صورت تصادفی یک جعبه‌ی جادویی به یکی از بازیکنان اختصاص داده می‌شود. دریافت این جعبه الزامی است و محتوای آن می‌تواند شامل یک طلسم یا یک هدیه باشد که بلافاصله بر وضعیت بازی تأثیر می‌گذارد.

محتوای جعبه‌ی جادویی به‌صورت تصادفی انتخاب شده و شامل یکی از موارد زیر است:

۱.۶ طلسم‌ها

طلسم‌ها اثر منفی بر وضعیت بازیکن یا کل صفحه‌ی بازی دارند و شامل موارد زیر هستند:

- حذف تمامی دیوارهای قرار داده‌شده در صفحه‌ی بازی
- کاهش دو، سه یا پنج دیوار از ظرفیت دیوارهای بازیکن
- بلاک شدن بازیکن برای یک یا دو نوبت متوالی

در صورت بلاک شدن، بازیکن در نوبت‌های مشخص‌شده اجازه‌ی انجام هیچ حرکتی (حرکت مهره یا قرار دادن دیوار) را نخواهد داشت.

۲.۶ هدیه‌ها

هدیه‌ها اثر مثبت بر وضعیت بازیکن دارند و شامل موارد زیر می‌باشند:

- افزایش دو، سه یا پنج دیوار به ظرفیت دیوارهای بازیکن
- کاهش یک یا دو دیوار از ظرفیت دیوارهای حریف و هم‌زمان افزایش ظرفیت دیوارهای بازیکن

اعمال اثر هر طلسم یا هدیه باید بلافاصله پس از دریافت آن انجام شود و تأثیر آن تا پایان مدت مشخص‌شده یا تا تغییر وضعیت مربوطه باقی می‌ماند. در صورتی که بازیکن در حالت بلاک قرار داشته باشد، جعبه‌ی جادویی همچنان به او اختصاص داده شده و اثر آن مطابق قوانین به‌صورت کامل اعمال می‌شود. در صورت هم‌زمان شدن چند اثر فعال، تمامی اثرها به‌ترتیب دریافت اعمال شده و هیچ اثری

باعث لغو اثر دیگر نخواهد شد. در صورت فعال شدن طلسم حذف دیوارها، تمامی دیوارهای موجود در صفحه حذف می‌شوند، اما ظرفیت دیوارهای بازیکنان تغییری نخواهد کرد و دیوارهای مصرف‌شده به بازیکن بازگردانده نمی‌شوند. در صورتی که در هر مرحله از بازی، پس از اعمال اثر یک طلسم یا هدیه، شرایط پیروزی برای یکی از بازیکنان فراهم شود، بازی بلافاصله پایان یافته و آن بازیکن برنده اعلام می‌شود.

۷ موارد نمره اضافه

علاوه بر بخشهای اصلی پروژه، پیاده‌سازی موارد زیر به‌عنوان نمره‌ی اضافه در نظر گرفته می‌شود. انجام هر یک از این موارد اختیاری بوده و نمره‌ی آن به‌صورت مستقل محاسبه خواهد شد.

۱. **بازی هوشمند**

پیاده‌سازی حالت بازی هوشمند برای بازیکن کامپیوتری، شامل نمره‌ی اضافه خواهد بود. هوشمندی بازیکن کامپیوتری می‌تواند در سطوح مختلف انجام شود:

- استفاده از الگوریتم‌های مسیریابی برای یافتن کوتاه‌ترین مسیر معتبر مانند Dijkstra، *A یا الگوریتم‌های مشابه، جهت تحلیل وضعیت صفحه‌ی بازی و انتخاب حرکت مناسب، شامل 3% نمره‌ی اضافه است.
- در صورتی که تصمیم‌گیری بازیکن کامپیوتری به‌صورت مجموعه‌ای از قوانین (قانون‌محور) پیاده‌سازی شود، به‌گونه‌ای که حرکت‌ها صرفاً تصادفی نباشند، 5% نمره‌ی اضافه تعلق می‌گیرد.
- در صورت استفاده از الگوریتم‌های تصمیم‌گیری پیشرفته‌تر مانند Min-Max، روش‌های مبتنی بر Monte Carlo (مانند Monte Carlo Simulation) یا الگوریتم‌های مشابه برای انتخاب حرکت بهینه، 10% نمره‌ی اضافه در نظر گرفته می‌شود.

۲. **بازی دو نفره تحت شبکه** پیاده‌سازی امکان بازی دو نفره تحت شبکه شامل 15% نمره‌ی اضافه خواهد بود.

۳. **بازی چهار نفره** پیاده‌سازی حالت بازی چهار نفره، به‌طوری که حداقل یک بازیکن توسط کاربر کنترل شده و سایر بازیکنان بتوانند توسط کامپیوتر یا کاربران دیگر کنترل شوند، شامل 15% نمره‌ی اضافه است.

۴. **استفاده از Git برای کنترل نسخه** استفاده از Git جهت مدیریت نسخه‌های پروژه و برنامه‌نویسی تیمی شامل 10% نمره‌ی اضافه خواهد بود.

۵. **پیاده‌سازی رابط کاربری گرافیکی** پیاده‌سازی رابط کاربری گرافیکی برای بازی شامل 15% نمره‌ی اضافه است.

۸ نکات

- پروژه باید در گروه‌های دو نفره پیاده‌سازی شود.
- شباهت کدها بین گروه‌ها بررسی خواهد شد و در صورت مشاهده‌ی تخلف، برای گروه‌های خاطی نمره‌ی 100- ثبت خواهد شد.

  شباهت کدها بین گروه‌ها با استفاده از ابزارهای تشخیص شباهت بررسی خواهد شد. در صورتی که میزان شباهت کد بین دو گروه بیش از 60% تشخیص داده شود، این وضعیت به‌عنوان تقلب تلقی شده و نمره‌ی پروژه برای هر دو گروه به‌صورت کامل 100- در نظر گرفته خواهد شد.

  مسئولیت رعایت اصالت کد به‌عهده‌ی تمامی اعضای گروه بوده و هرگونه استفاده‌ی غیرمجاز از کد سایر گروه‌ها یا منابع بدون ارجاع مناسب، مشمول این قانون خواهد بود.
- از مجموع موارد نمره‌ی اضافه، حداکثر 50% قابل دریافت است.
- تمیزی کد و میزان تسلط اعضای گروه بر پروژه‌ی پیاده‌سازی‌شده، به‌عنوان ضریبی از نمره‌ی نهایی پروژه در نظر گرفته می‌شود. به این صورت که در هنگام تحویل، در صورتی که تسلط فردی کمتر از 90% ارزیابی شود، این مقدار به‌عنوان ضریب از نمره‌ی دریافتی کسر خواهد شد.
- در پیاده‌سازی پروژه، استفاده از struct و enum در بخش‌های مناسب الزامی است. به‌عنوان مثال، اطلاعات مربوط به هر خانه از جدول بازی می‌تواند با استفاده از یک struct نگهداری شود و وضعیت هر خانه (مانند خالی بودن، وجود مهره یا وجود دیوار) باید با استفاده از یک enum مدل‌سازی گردد.
- در ارتباط با پیاده‌سازی پروژه به‌صورت گرافیکی (به‌عنوان نمره‌ی اضافه)، می‌توان از کتابخانه‌های raylib و allegro استفاده کرد. برای یادگیری raylib، مراجعه به وب‌سایت https://www.raylib.com و بررسی بخش examples پیشنهاد می‌شود. همچنین برای یادگیری allegro می‌توان از مستندات رسمی این کتابخانه در آدرس https://liballeg.org/docs.html استفاده کرد.
- تعدادی نکته و راهنمای تکمیلی در فایل پیوست ارائه شده است که دانشجویان می‌توانند از آن‌ها در پیاده‌سازی پروژه استفاده نمایند.
- ارائه و ارزیابی پروژه به‌صورت یکپارچه و در قالب یک تحویل نهایی انجام خواهد شد. تقسیم‌بندی پروژه به فازهای مختلف، صرفاً با هدف کمک به فرآیند پیاده‌سازی مرحله‌به‌مرحله در طول انجام پروژه ارائه شده است. ملاک و معیار نمره‌دهی نهایی، کیفیت کلی پیاده‌سازی، صحت عملکرد برنامه، رعایت قوانین پروژه و میزان تسلط اعضای گروه بر کل پروژه خواهد بود، نه تحویل یا ارزیابی جداگانه‌ی هر فاز.

۹ معیارهای نمره‌دهی پروژه

نمره‌ی پایه‌ی پروژه از 100 محاسبه می‌شود. ارزیابی پروژه به‌صورت یکپارچه انجام شده و تقسیم‌بندی فازها صرفاً جهت هدایت فرآیند پیاده‌سازی بوده است.

۱.۹ فاز اول: ایجاد نقشه و نمایش بورد (10 نمره)

- دریافت و پردازش صحیح ورودی‌ها (ابعاد، بازیکنان، دیوارها): 3 نمره
- نمایش بورد بازی با استفاده از کاراکترهای ASCII: 4 نمره
- نمایش دقیق موقعیت بازیکنان و دیوارها مطابق ورودی: 2 نمره
- خوانایی و نظم خروجی: 1 نمره

۲.۹ فاز دوم: منطق بازی دو نفره (30 نمره)

- پیاده‌سازی صحیح نوبت‌بندی بازیکنان: 5 نمره
- منطق صحیح حرکت مهره مطابق قوانین: 7 نمره
- پیاده‌سازی صحیح دیوارگذاری و محدودیت‌ها: 8 نمره
- جلوگیری از حرکات و دیوارگذاری نامعتبر: 5 نمره
- تشخیص پایان بازی و اعلام برنده: 5 نمره

۳.۹ فاز سوم: بازی با کامپیوتر (15 نمره)

- انتخاب حالت بازی (کاربر ـ کاربر / کاربر ـ کامپیوتر): 3 نمره
- تصمیم‌گیری تصادفی کامپیوتر با استفاده از random: 4 نمره
- رعایت کامل قوانین بازی توسط کامپیوتر: 5 نمره
- مدیریت تصمیم‌های نامعتبر و انتخاب مجدد: 3 نمره

۴.۹ فاز چهارم: ذخیره و بازی مجدد (15 نمره)

- ذخیره‌ی کامل و صحیح وضعیت بازی در فایل: 6 نمره
- بارگذاری صحیح بازی ذخیره‌شده و ادامه از همان وضعیت: 6 نمره
- مدیریت خطاها و سناریوهای ورودی نامعتبر: 3 نمره

۵.۹ فاز پنجم: طلسم‌ها و پاداش‌ها (30 نمره)

- پیاده‌سازی جعبه‌ی جادویی و انتخاب تصادفی محتوا: 6 نمره
- اعمال صحیح اثر طلسم‌ها: 8 نمره
- اعمال صحیح اثر هدیه‌ها: 8 نمره
- مدیریت بلاک شدن بازیکن و تأثیر آن بر روند بازی: 4 نمره
- هماهنگی کامل این فاز با منطق کلی بازی: 4 نمره

جمع نمره‌ی پایه

بخش ارزیابی	نمره
فاز اول	۱۰
فاز دوم	۳۰
فاز سوم	۱۵
فاز چهارم	۱۵
فاز پنجم	۳۰
جمع کل	**۱۰۰**

۶.۹ ضریب تمیزی کد و تسلط

پس از محاسبه‌ی نمره‌ی پایه، میزان تمیزی کد و تسلط اعضای گروه بر پروژه به‌عنوان ضریب نهایی بر نمره‌ی به‌دست‌آمده اعمال می‌شود.

در صورتی که میزان تسلط فردی هر یک از اعضای گروه در هنگام ارائه کمتر از 100% ارزیابی شود، نمره‌ی نهایی پروژه متناسب با این میزان کاهش خواهد یافت. نمره‌ی نهایی پروژه از رابطه‌ی زیر محاسبه می‌شود:

$$\text{نمره نهایی} = \text{نمره پایه} \times \text{ضریب تسلط}$$

۱۰ نکات تکمیلی

- **رابط متنی پیشرفته** در صورت تمایل به پیاده‌سازی رابط کاربری متنی پیشرفته‌تر، می‌توان از کتابخانه‌ی Ncurses استفاده کرد. اطلاعات بیشتر و مستندات این کتابخانه از طریق

  مستندات رسمی Ncurses

  در دسترس می‌باشد.

- **تولید اعداد تصادفی** برای تولید اعداد تصادفی می‌توان از تابع ()rand استفاده کرد. از آن‌جایی که خروجی این تابع به‌صورت پیش‌فرض در اجرای‌های مختلف برنامه یکسان است، لازم است در ابتدای اجرای برنامه و تنها یک‌بار، تابع ((srand(time(NULL فراخوانی شود تا دنباله‌ی اعداد تصادفی در هر اجرای برنامه متفاوت باشد. پس از این مقداردهی اولیه، هر بار فراخوانی تابع ()rand منجر به تولید اعداد تصادفی متفاوت خواهد شد.

- **مدیریت نسخه با Git** در پیاده‌سازی پروژه، استفاده از Git برای کنترل نسخه توصیه می‌شود. پیشنهاد می‌شود از الگوی Git Flow استفاده گردد (مطالعه از این منبع). در این الگو، شاخه‌ی master/main باید همواره شامل نسخه‌ی پایدار پروژه باشد، به‌گونه‌ای که در هر زمان با clone کردن مخزن، امکان build و اجرای برنامه بدون مشکل وجود داشته باشد.

  همچنین انتخاب پیام‌های مناسب برای commitها اهمیت زیادی دارد. راهنمای نگارش پیام commit را می‌توانید از این منبع مطالعه کنید.

  لازم است فایل‌های اضافی و خروجی‌های کامپایل مانند فایل‌های با پسوند exe، o، obj و موارد مشابه در فایل gitignore. قرار گیرند تا در commitها لحاظ نشوند.

- **کدنویسی تمیز و ساختارمند** رعایت اصول Clean Code در پیاده‌سازی پروژه الزامی است. راهنمایی‌هایی در این زمینه از طریق این منبع قابل مطالعه می‌باشد.

  پروژه باید از چندین فایل مجزا تشکیل شده باشد و از نوشتن تمام کد در یک فایل خودداری شود. همچنین استفاده از توابع متعدد، کوتاه و خوانا توصیه می‌شود. هر تابع باید تنها مسئول انجام یک وظیفه‌ی مشخص باشد و از واگذاری چندین مسئولیت به یک تابع اجتناب گردد. برای افزایش خوانایی کد، استفاده‌ی مناسب از کامنت‌ها الزامی است.

موفق باشید

تیم حل تمرین مبانی کامپیوتر و برنامه‌سازی

پاییز ۱۴۰۴