**Proyecto. Valor: 35%.**

**Curso: Programación III.**

**Docente: Alejandro Alfaro Quesada. (aalfaroq@utn.ac.cr)**

**Fecha de entrega: martes 06 de julio del 2021.**

**Hora de entrega: 05:00 pm.**

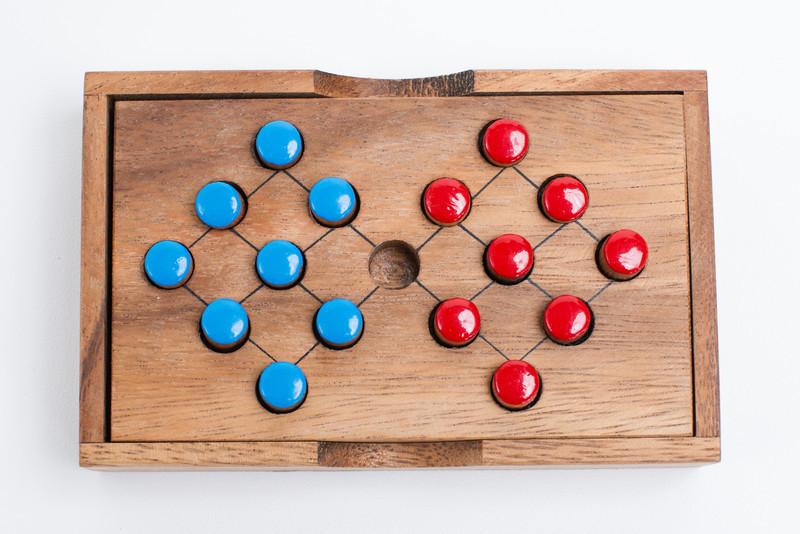
**Indicaciones Generales.**

* La prueba se desarrolla bajo la modalidad de proyecto, con tiempo de inicio y fin predeterminado.
* Toda prueba evaluada es considerada como un único proyecto, es decir, aunque la evaluación sea dividida por secciones, la completitud de uno o más rubros puede afectar la funcionalidad del sistema, y esto, podría influir en la nota final.
* La prueba puede ser desarrollada en un máximo de dos personas. Para los que están conformados en parejas, se debe contemplar la siguiente regla:
  + Si por alguna razón no puedan continuar trabajando juntos, ambos deben continuar con el código fuente que tienen hasta ese momento; es decir, no se permite que se unan a otro compañero y/o utilizar el código fuente de otro compañero.
* Si algún estudiante es sorprendido en actos fraudulentos, ya sea que los provoque o los consienta, su prueba y la de los demás implicados, será automáticamente anulada.
* El docente ejecuta el código fuente que está en el proyecto enviado para comprobar que realiza lo solicitado. Por lo que, el docente NO modificará ninguna línea de código, por lo tanto, debe asegurarse que el código fuente funciona correctamente.
* El proyecto con el código solo se recibe y se valida desde el Campus Virtual de la UTN. Por cualquier otro medio que se envíe al docente no será revisado.

**Indicaciones Generales.**

* En el proyecto debe utilizar programación por capas, objetos, base de datos, estructuras de datos e interfaz gráfica obligatoriamente.

**Cambio de Fichas**



**Secciones del proyecto**

Se puede observar el video de ejemplo en el siguiente enlace, a partir del minuto 7 en adelante.

https://www.youtube.com/watch?v=PEJyEIyFe1I

**Pantalla de inicio**

Debe haber una pantalla de inicio, en donde deben aparecer los siguientes elementos:

1. Su nombre completo (se colocan todos los miembros del equipo).
2. Una fotografía suya (debe estar dentro del proyecto y se coloca todas las de los miembros del equipo).
3. La fecha y hora actual del sistema.
4. Un mensaje de bienvenida al juego, además se le solicita al usuario ingresar el nombre (se va a asumir que no se van a ingresar nombres repetidos).
5. Un botón que permite acceder a la segunda ventana del sistema.

**Resolución del juego**

* El tablero debe estar representado en la interfaz tal y como se muestra en la imagen de ejemplo.
* El sistema no puede limitarse a una única manera de resolver el juego, si no, que, mediante las reglas del juego, el usuario debe analizar los movimientos.
* El juego tiene las siguientes reglas que se deben respetar en la programación:
  + Las piezas se pueden mover solamente hacia adelante.
  + ~~Las piezas se mueven una a la vez, indiferentemente del color.~~
  + ~~Se puede hacer un único movimiento a la vez, de cada pieza de manera adyacente.~~
  + Se puede hacer uno o más movimientos a la vez, de cada color de pieza de manera adyacente.
  + Se puede saltar una única vez cada pieza, siempre y cuando sea de diferente color.
  + El usuario no puede mover una pieza hacia atrás, ni, aunque sea adyacente ni que sea un salto.
* El juego se puede reiniciar cada vez que el usuario lo considere necesario.
* Durante cada partida de juego por usuario, se debe ir controlando los siguientes aspectos:
  + Cantidad de movimientos adyacentes, independientemente del color de la pieza que se mueva.
  + Cantidad de saltos, independientemente del color de la pieza que se esté saltando.
* El juego se puede catalogar como resuelto, si cumple con todas las reglas y si todas las piezas de ambos colores cambiaron de posición de un lado al otro.

**Base de datos**

* Cada vez, que el juego se reinicie o el usuario logre encontrar la solución, se debe guardar la siguiente información en la base de datos.
  + Nombre del usuario.
  + Fecha y hora (cuando se registró para iniciar el juego).
  + Cantidad de movimientos adyacentes.
  + Cantidad de saltos.
  + Un campo que sea para el tipo de juego ya sea que lo reinició o que encontró la solución.
* Algunos ejemplos de la información que se guarda:
  + Alejandro | 12/02/2021 6:00 PM | 33 | 12 | Reinicio
  + Alejandro | 12/02/2021 6:32 PM | 21 | 12 | Reinicio
  + Alejandro | 14/02/2021 7:32 PM | 40 | 18 | Resuelto

**Reportes**

* Se deben crear los siguientes reportes utilizando un DataGridView para mostrar la información:
  1. Menor cantidad de movimientos adyacentes hecho por un único usuario. Debe mostrar el nombre, fecha, hora y la cantidad de movimientos.
  2. Cantidad de veces que se resolvió el juego por cada participante. Debe mostrar el nombre de cada participante y la cantidad de veces que lo resolvió. Tome en cuenta que es un único registro que se muestra, independientemente el usuario lo haya resuelto muchas veces.
  3. Cantidad de veces que un único usuario reinició el juego. Se hace la búsqueda por una persona en específico. Debe mostrar el nombre y la cantidad de reinicios. Tome en cuenta que es un único registro que se muestra, independientemente el usuario lo haya reiniciado muchas veces.

**Dashboard**

* Se deben crear un Dashboard que estará después de la ventana de inicio, en donde a través de gráficos se representa lo que se solicita:

1. Gráfico de pastel con la relación de cantidad de resueltos vs cantidad de reinicios del juego.

Gráfico, Gráfico circular

Descripción generada automáticamente

1. Gráfico de barras con el top 3 de las personas que han resuelto más veces el juego.

Gráfico, Gráfico de barras

Descripción generada automáticamente

1. Gráfico de barras con el top 3 de los juegos resueltos con la menor cantidad de movimientos adyacentes.

Gráfico, Gráfico de barras

Descripción generada automáticamente

**Aspectos generales.**

**Documentación externa.**

* Debe crear un documento con una portada y donde explique cómo resolvió el juego. Toda la explicación debe estar redactada en inglés.
* Además, incluya las dos siguientes tablas en el documento.

Tabla de completitud.

|  |  |  |
| --- | --- | --- |
| **Rubro** | **Estado %** | **Observaciones** |
| **Secciones del juego** |  |  |
| * Pantalla de inicio | Completo (100%) |  |
| * Solución del juego | Incompleto (70%) | Faltan condiciones. |
| * Archivos |  |  |
| * Reportes | Incompleto (40%) | Faltan 2 reportes. |

Porcentaje de participación.

|  |  |  |
| --- | --- | --- |
| **Rubro** | **María** | **Carlos** |
| **Secciones del juego** |  |  |
| * Pantalla de inicio | 40% | 60% |
| * Solución del juego | 70% | 30% |
| * Archivos | 50% | 50% |
| * Reportes | 40% | 60% |

**Entregables.**

A continuación, se lista lo que debe entregar como un todo, es decir, el faltante de alguno implica que no está entregando el proyecto completo.

* Proyecto programado.
* Scripts y backup de la base de datos.
* Archivo txt con el enlace del repositorio de Git.
* Documentación externa completa, es decir, portada, redacción en inglés de la solución del juego, además, las dos tablas que están en el apartado anterior.