Cégep de Saint-Hyacinthe
Département d'informatique

Algorithme et programmation (420-1MP-HY)

Travail Pratique #3 (Jeu de Simon)

Préparé par
Martin Lalancette
bureau B-2230

DESCRIPTION

But :	Utiliser des tableaux 1D et des méthodes en renforçant les autres notions apprises précédemment.
Objectifs :	• Appliquer les principes logiques algorithmiques avec une bonne syntaxe en suivant vigoureusement les **normes de programmation** • Pratiquer les structures alternatives et itératives, etc. • Pratiquer les notions de tableau 1D • Créer et utiliser des méthodes. • Programmer une application de type console • Être capable d'utiliser l'outil MS Visual Studio.
Durée :	6 h 00
Pondération :	sur 10
Remise :	À la fin de la semaine 12.
Contenu général :	• Remettre vos documents d'analyse et votre solution contenant le projet dans un document .ZIP via LÉA.
Notes	• Conserver une copie de sécurité. Il est de **votre responsabilité** de conserver une copie de sécurité dans l'éventualité où la lecture des données serait impossible. Cette copie doit **être disponible sur demande**.

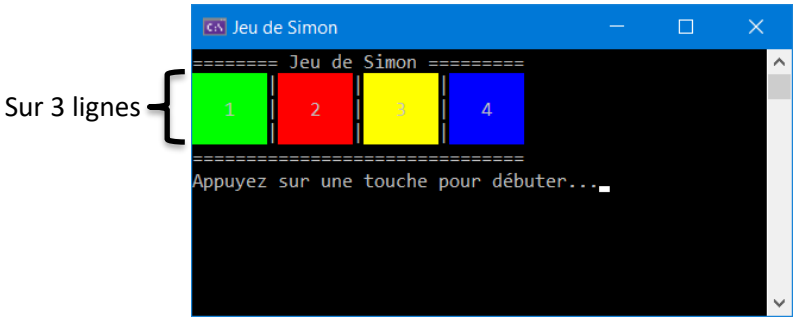
Jeu de Simon (projet console) – 100%

Vous devez concevoir une application de type console basée sur le jeu de Simon simplifié. Le jeu consiste à allumer une couleur, puis ajouter au hasard une nouvelle couleur par la suite. Le joueur doit reproduire cette nouvelle séquence. Chaque fois que le joueur reproduit correctement la séquence, le jeu ajoute une nouvelle couleur supplémentaire.

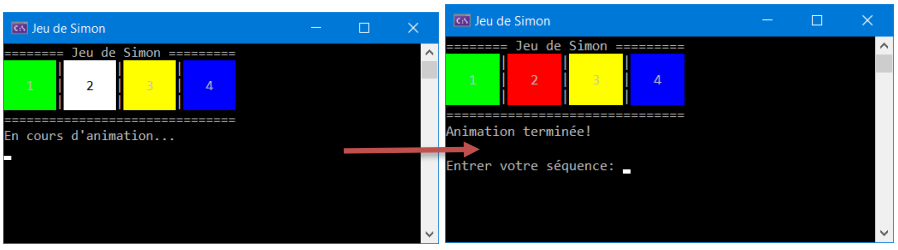
Lien vidéo : https://web.microsoftstream.com/video/30d3c46b-edbc-4bb0-9545-cc365006e0b1

Stratégies :

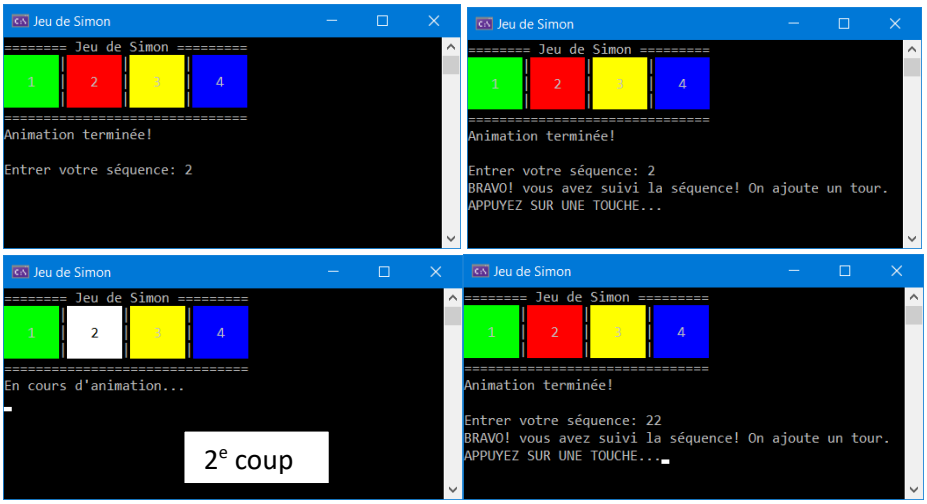
1. Créer un algorithme qui va générer une séquence de 255 coups à l'avance. **Vous devez utiliser un tableau 1D de type « byte »** pour contenir les 255 coups. L'objet Random() vous sera utile pour bâtir cette séquence.
2. Voici un exemple d'interface à produire:



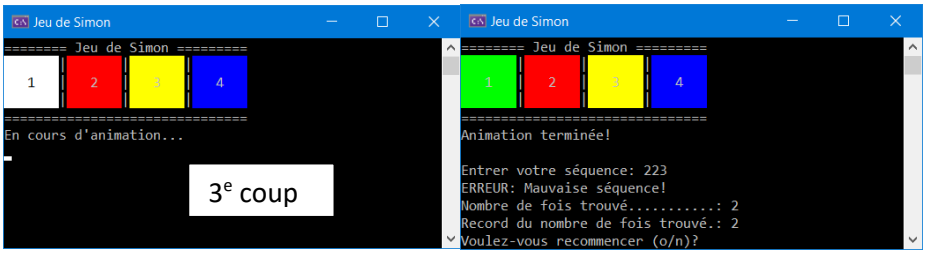
   a. **Couleurs**: 1=Green, 2=Red, 3=Yellow et 4=Blue. Utiliser Console.BackgroundColor.
3. Voici une idée sommaire du fonctionnement du programme:
   a. Lorsque l'utilisateur démarre la console, le programme doit générer une séquence de 255 coups d'avance (cette étape est transparente à l'usager), afficher les 4 couleurs et attendre que l'usager appuie sur une touche. Par la suite, le programme doit animer (changer de couleur) le premier coup et attendre l'utilisateur. Exemple :



   b. L'utilisateur doit saisir la séquence depuis le début et faire ENTRER. Le programme doit valider cette séquence et indiquer s'il y a erreur ou non. Si la séquence est bonne, le programme doit ajouter une couleur supplémentaire et animer du début jusqu'à la nouvelle couleur. Exemples :



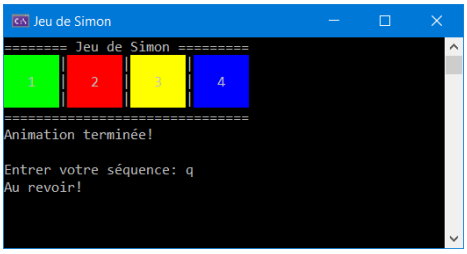
À répéter en ajoutant un coup supplémentaire tant aussi longtemps que l'utilisateur ne fait pas d'erreur. Si l'utilisateur se trompe, un message d'erreur apparaît et les statistiques apparaissent également. Exemple :



Par la suite, le programme génère une nouvelle séquence de 255 coups et recommence au premier coup.

c. Vous devez conserver les coups de l'utilisateur afin de comparer avec ceux qui ont été générés.

d. En tout temps, si l'utilisateur souhaite quitter le programme, il peut saisir la lettre Q (minuscule et majuscule) durant une partie. Exemple :



e. Ajuster les pointages en conséquence :

   i. **Nombre de fois trouvé** : est le nombre de coups réussis durant une partie.

   ii. **Record du nombre de fois trouvé** : est le nombre de coups réussis le plus élevé parmi toutes les parties jouées.

4. À savoir :

   a. Pour laisser le temps à l'utilisateur de voir les changements de couleur, il faut effectuer des pauses avec un certain délai. Pour effectuer cette pause, il faut

utiliser `System.Threading.Thread.Sleep(1000);` Ici, le 1000 représente 1000 millisecondes donc 1 seconde. Vous pouvez utiliser le *using* pour simplifier.

Consignes :

1. Découper votre algorithme en plusieurs méthodes afin de diminuer le nombre de lignes contenu dans le programme principal et bien structurer votre code.
2. Créer une solution nommée **TP3** et ajouter le projet de type console (.NET Core) nommé **JeuSimon.**
3. Ajouter un fichier texte (ou word) dans ce projet et y **rédiger le pseudo-code**.
4. Concevoir **tous les cas de test** reliés à la **validation et robustesse** du programme en utilisant la grille de tests dans LÉA.
5. Dans le fichier **Program.cs**, coder la logique de votre pseudo-code en ajoutant des commentaires (Description, Auteur, Date, etc.) et en appliquant les normes**.**
6. Effectuer **tous les cas** de test pour s'assurer de la bonne fonctionnalité du programme.

Barème d'évaluation

Pseudo-code (Découpage en méthodes, etc.) ..... /1.5
Élaboration et exécution des cas de tests (GrilleDeTests.xlxs) ..... /1.5
Programmation : Séquence des lumières – hasard et animation ..... /2.0
Programmation : Validation de la saisie de la séquence ..... /2.0
Programmation : Fonctionnement des statistiques ..... /1.0
Programmation : Gestion des erreurs ..... /1.0
Programmation : Fonctionnement général et structure (ex. gestion des parties, méthodes) ... /1.0
Note totale* ..... /10.0

* Tout travail plagié en partie ou en totalité se verra attribuer une note totale de 0 %.

PDEA #1: Lors d'activités d'évaluation sommative en classe ou hors classe (documentation, rapport de laboratoire, rapport de stage, examen), une **pénalité maximale de 10 %** peut être retranchée de la note finale de ladite évaluation (le barème étant de **0,5%/erreur incluant les fautes répétitives**). Pour les commentaires dans les programmes informatiques, le barème est de **0,5% par faute**. Pour les interfaces utilisateur, le barème est de **1% par faute**.

PDEA #3 : Toute évaluation sommative remise **après la date d'échéance fixée** se voit **attribuer la note zéro**.