**Programmation structurée**

**tITRE du CoursT**

**420-P24-MW**

**Travail pratique No. 1**



**DA : 2230017**

**Nom : Couture**

**Prénom : Maryève**

**DA : 2230002**

**Nom : Allard**

**Prénom : Justin**

**Date : 2023-01-10**

Faites un programme permettant de simuler des tirages de 6/49.  
  
Dans un premier temps, demander à votre utilisateur le nombre de combinaisons que celui-ci désire (entre 10 et 200 combinaisons). Générer et afficher les combinaisons de l’utilisateur.  
  
Par la suite, faites générer la combinaison gagnante de 6 chiffres avec le chiffre complémentaire.  
  
Enfin indiquer, les combinaisons gagnantes de l’utilisateur.

Voici le tableau des combinaisons qui ne donnent aucun gain :

* 0 sur 6
* 1 sur 6
* 2 sur 6

Voici le tableau des combinaisons qui donnent des gains :

* 3 sur 6
* 3 sur 6 +
* 4 sur 6
* 4 sur 6 +
* 5 sur 6
* 5 sur 6 +
* 6 sur 6

A la fin de votre programme, sortez les statistiques indiquant le nombre de tirages et le nombre d’apparitions de chaque nombre a eu à l’intérieur des combinaisons gagnantes, ainsi que le % de combinaisons pour chaque famille de résultats.

**ATTENTION**

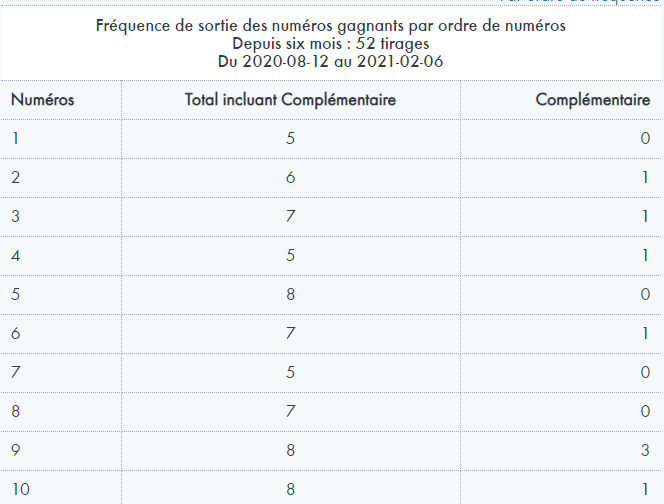
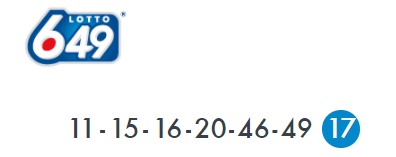
**\*\*** Votre programme devra être monté avec l’aide de procédures et des fonctions.  
\*\* **À chaque fois que votre programme se termine, votre programme devra demander à l’utilisateur s’il veut refaire un tirage ou non.**

# À remettre

- Ce travail peut être fait en équipe de 2 personnes.

* Fichier compressé contenant votre projet C# via le devoir de Teams.

|  |
| --- |
| **Ce Travail pratique est à remettre au plus tard**  **La journée de l’examen intra à 16h30.** |



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\* PROGRAMME - CRÉATION DE COMBINAISONS DE 6/49 \*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Description : Ce programme demande à l'utilisateur un nombre de

\* combinaisons de 6/49 entre 10 et 200, tire au hasard les

\* combinaisons de 6/49, les chiffres dans chaque combinaison doivent

\* être sans répétition, avec un numéro complémentaire et doit afficher

\* la combinaison dans l'ordre (le numéro complémentaire ne doit pas

\* être dans l'ordre).

\*

\* Par la suite, une combinaison gagnante avec le chiffre complémentaire

\* sera généré

\*

\* Indiquer les combinaisons gagnantes de l'utilisateur.

\*

\* À la fin du programme, il y aura :

\* - des statistiques indiquant le nombre de tirages

\* - le nombre d'apparition de chaque nombre à l'intérieur des

\* combinaisons gagnantes

\* - Le pourcentage de combinaisons pour chaque famille de résultat

\*

\* Fait par: Marie-Ève couture

\* Fait par: Justin Allard

\* Le : 2023-01-09

\* Révisé par :

\* Révisé le :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace T24\_TP1\_MEC\_JA

{

// CRÉATION DE LA STRUCTURE

struct Combi\_Gagnante\_Perdante

{

public int[] Combi\_Perdantes;

public int[] Combi\_3\_6;

public int[] Combi\_3\_6\_C;

public int[] Combi\_4\_6;

public int[] Combi\_4\_6\_C;

public int[] Combi\_5\_6;

public int[] Combi\_5\_6\_C;

public int[] Combi\_Gagnante;

}

internal class Program

{

static void Main(string[] args)

{

// MESSAGE D'ACCUEIL

Message\_Accueil();

int nombre\_tirage = 0;

int[] comparaison = new int[49];

do

{

// SECTION UTILISATEUR

int nombre\_combinaisons = Demander\_Nombre\_Combinaisons(10, 200);

int[,] Combinaison\_Utilisateur = new int[nombre\_combinaisons, 7];

int[,] tableau\_generer\_utilisateur = Generer\_Combinaisons\_Utilisateur(Combinaison\_Utilisateur, nombre\_combinaisons);

int[,] tableau\_trier\_utilisateur = Trier\_Combinaison\_Utilisateur(tableau\_generer\_utilisateur);

int[,] combinaison\_finale\_utilisateur = Afficher\_Tableau\_Utilisateur(tableau\_trier\_utilisateur);

// SECTION GÉNÉRATEUR DE LOTERIE

int[] combinaison\_gagnante = Generer\_Combinaison\_Gagnante();

int[] combinaison\_trier\_gagnante = Trier\_Combinaison\_Gagnante(combinaison\_gagnante);

int[] combinaison\_finale\_gagnante = Afficher\_Combinaison\_Gagnante(combinaison\_trier\_gagnante);

foreach (int numeroGagnant in combinaison\_gagnante)

{

comparaison[numeroGagnant - 1]++;

}

nombre\_tirage++;

// CLASSEMENT DES COMBINAISONS DANS LES BONNES FAMILLES

Combi\_Gagnante\_Perdante tirage = new Combi\_Gagnante\_Perdante()

{

Combi\_Perdantes = Array.Empty<int>(),

Combi\_3\_6 = Array.Empty<int>(),

Combi\_3\_6\_C = Array.Empty<int>(),

Combi\_4\_6 = Array.Empty<int>(),

Combi\_4\_6\_C = Array.Empty<int>(),

Combi\_5\_6 = Array.Empty<int>(),

Combi\_5\_6\_C = Array.Empty<int>(),

Combi\_Gagnante = Array.Empty<int>()

};

for (int i = 0; i < combinaison\_finale\_utilisateur.GetLength(0); i++)

{

int nbGagnant = 0;

bool avecComplementaire = false;

for (int j = 0; j < combinaison\_finale\_utilisateur.GetLength(1); j++)

{

if (Array.IndexOf(combinaison\_finale\_gagnante, combinaison\_finale\_utilisateur[i, j]) >= 0)

{

nbGagnant++;

}

}

switch (nbGagnant)

{

case 0:

case 1:

case 2:

tirage.Combi\_Perdantes = tirage.Combi\_Perdantes.Append(i).ToArray();

break;

case 3:

if (avecComplementaire)

tirage.Combi\_3\_6\_C = tirage.Combi\_3\_6\_C.Append(i).ToArray();

else

tirage.Combi\_3\_6 = tirage.Combi\_3\_6.Append(i).ToArray();

break;

case 4:

if (avecComplementaire)

tirage.Combi\_4\_6\_C = tirage.Combi\_4\_6\_C.Append(i).ToArray();

else

tirage.Combi\_4\_6 = tirage.Combi\_4\_6.Append(i).ToArray();

break;

case 5:

if (avecComplementaire)

tirage.Combi\_5\_6\_C = tirage.Combi\_5\_6\_C.Append(i).ToArray();

else

tirage.Combi\_5\_6 = tirage.Combi\_5\_6.Append(i).ToArray();

break;

case 6:

tirage.Combi\_Gagnante = tirage.Combi\_Gagnante.Append(i).ToArray();

break;

}

}

// STATISTIQUES DES FAMILLES

AfficherCombiConcordantes("3 sur 6", combinaison\_finale\_utilisateur, tirage.Combi\_3\_6, nombre\_combinaisons);

AfficherCombiConcordantes("3 sur 6 avec no complémentaire", combinaison\_finale\_utilisateur, tirage.Combi\_3\_6\_C, nombre\_combinaisons);

AfficherCombiConcordantes("4 sur 6", combinaison\_finale\_utilisateur, tirage.Combi\_4\_6, nombre\_combinaisons);

AfficherCombiConcordantes("4 sur 6 avec no complémentaire", combinaison\_finale\_utilisateur, tirage.Combi\_4\_6\_C, nombre\_combinaisons);

AfficherCombiConcordantes("5 sur 6", combinaison\_finale\_utilisateur, tirage.Combi\_5\_6, nombre\_combinaisons);

AfficherCombiConcordantes("5 sur 6 avec no complémentaire", combinaison\_finale\_utilisateur, tirage.Combi\_5\_6\_C, nombre\_combinaisons);

AfficherCombiConcordantes("gagnantes", combinaison\_finale\_utilisateur, tirage.Combi\_Gagnante, nombre\_combinaisons);

} while (Recommencer());

// MESSAGE DU NOMBRE DE TIRAGES

Console.ResetColor();

Console.Write("Le nombre de tirage est de {0}", nombre\_tirage);

// COMPARAISON DES NOMBRES GAGNANTS

Comparaison\_Nombre\_Gagnant(comparaison);

// MESSAGE DE FIN DE PROGRAMME

Message\_Fin\_Programme();

}

#region 1 - MESSAGE D'ACCUEIL

/// <summary>

/// Procédure Message d'accueil à l'utilisateur

/// </summary>

public static void Message\_Accueil()

{

Console.SetCursorPosition(0, 1);

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("BIENVENUE SUR LE GÉNÉRATEUR DE NUMÉROS DE LOTERIE !");

Console.ForegroundColor = ConsoleColor.White;

Console.SetCursorPosition(0, 3);

Console.WriteLine("Veuillez appuyer sur ENTRÉE à chaque étape afin de saisir les informations.");

}

#endregion

#region 2 - DEMANDER NOMBRE COMBINAISONS UTILISATEUR

/// <summary>

/// Fonction permettant à l'utilisateur de déterminer le nombre de combinaisons voulues, entre 10 et 200

/// </summary>

/// <param name="min"></param>

/// <param name="max"></param>

/// <returns>Un integer du nombre de combinaisons</returns>

public static int Demander\_Nombre\_Combinaisons(int min, int max)

{

// DEMANDER À L'UTILISATEUR LE NOMBRE DE COMBINAISONS

int nombre\_combinaisons;

while (true)

{

Console.SetCursorPosition(0, 5);

Console.ForegroundColor = ConsoleColor.White;

Console.Write("Veuillez entrer le nombre de combinaisons désiré (entre 10 et 200): ");

int.TryParse(Console.ReadLine(), out nombre\_combinaisons);

Console.WriteLine();

if (nombre\_combinaisons >= min && nombre\_combinaisons <= max)

return nombre\_combinaisons;

else

Console.WriteLine("Attention! Vous devez entrer un nombre de combinaisons entre 10 et 200 !");

}

}

#endregion

#region 3 - GÉNÉRER COMBINAISON UTILISATEUR

/// <summary>

/// Procédures permettant de demander à l'utilisateur le nombre de combinaisons voulues, de trier et d'afficher ces combinaisons

/// </summary>

/// <param name="tableau"></param>

/// <param name="max"></param>

public static int[,] Generer\_Combinaisons\_Utilisateur(int[,] tableau, int max)

{

// GÉNÉRER LES COMBINAISONS DE L'UTILISATEUR

Random hasard = new Random();

bool[] numero\_sorti = new bool[49];

int numero = 0;

for (int i = 0; i <= tableau.GetUpperBound(0); i++)

{

Array.Clear(numero\_sorti, 0, numero\_sorti.Length);

for (int j = 0; j <= tableau.GetUpperBound(1); j++)

{

do

{

numero = hasard.Next(49);

} while (numero\_sorti[numero] == true);

numero\_sorti[numero] = true;

tableau[i, j] = numero + 1;

}

}

return tableau;

}

public static int[,] Trier\_Combinaison\_Utilisateur(int[,] tableau)

{

// TRIER LES COMBINAISONS DE L'UTILISATEUR

for (int i = 0; i < tableau.GetLength(0); i++)

{

for (int j = tableau.GetLength(1) - 1; j > 0; j--)

{

for (int k = 0; k < j - 1; k++)

{

if (tableau[i, k] > tableau[i, k + 1])

{

int temporaire = tableau[i, k];

tableau[i, k] = tableau[i, k + 1];

tableau[i, k + 1] = temporaire;

}

}

}

}

return tableau;

}

public static int[,] Afficher\_Tableau\_Utilisateur(int[,] tableau)

{

// AFFICHER LA COMBINAISON

Console.BackgroundColor = ConsoleColor.Black;

Console.ForegroundColor = ConsoleColor.White;

for (int i = 0; i <= tableau.GetUpperBound(0); i++)

{

Console.Write($"Votre combinaison #{i + 1}: ");

for (int j = 0; j < tableau.GetUpperBound(1); j++)

{

Console.Write(tableau[i, j].ToString().PadLeft(2, '0'));

Console.Write(" ");

}

Console.WriteLine();

}

Console.ReadLine();

return tableau;

}

#endregion

#region 4 - TIRAGE DU NUMÉRO GAGNANT

/// <summary>

/// Fonctions permettant de générer, de trier et d'afficher un numéro gagnant avec le complémentaire (Complémentaire non trié)

/// </summary>

/// <returns>La combinaison gagnante</returns>

public static int[] Generer\_Combinaison\_Gagnante()

{

// CRÉATION DU TABLEAU

int[] combinaison = new int[7];

Random hasard = new Random();

bool[] numero\_sorti = new bool[49];

int numero;

// RÉINITIALISATION DU TABLEAU

Array.Clear(numero\_sorti, 0, numero\_sorti.Length);

// TIRAGE AU HASARD DES NUMÉROS

for (int i = 0; i < combinaison.Length; i++)

{

do

{

numero = hasard.Next(49);

} while (numero\_sorti[numero] == true);

numero\_sorti[numero] = true;

combinaison[i] = numero + 1;

}

return combinaison;

}

public static int[] Trier\_Combinaison\_Gagnante(int[] combinaison)

{

// TRIER LA COMBINAISON DE 6 CHIFFRES SANS TRIER LE COMPLÉMENTAIRE

int tempo;

for (int i = 0; i < combinaison.Length - 1; i++)

{

for (int j = i + 1; j < combinaison.Length - 1; j++)

{

if (combinaison[i] > combinaison[j])

{

tempo = combinaison[i];

combinaison[i] = combinaison[j];

combinaison[j] = tempo;

}

}

}

return combinaison;

}

public static int[] Afficher\_Combinaison\_Gagnante(int[] combinaison)

{

// AFFICHER LA COMBINAISON GAGNANTE

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("La combinaison ");

Console.Write("gagnante est : \t\t");

for (int i = 0; i < combinaison.Length; i++)

{

if (i == 6)

{

Console.Write(" Complémentaire: ");

}

Console.Write(combinaison[i].ToString().PadLeft(2, '0'));

Console.Write(" ");

}

Console.ResetColor();

Console.ReadLine();

Console.Clear();

return combinaison;

}

#endregion

#region 5 - STATISTIQUES

/// <summary>

/// Procédure permettant de comparer et d'afficher les statistiques

/// </summary>

/// <param name="combinaison\_gagnante"></param>

public static void Comparaison\_Nombre\_Gagnant(int[] combinaison\_gagnante)

{

Console.Clear();

Console.WriteLine("Le nombre de numéros sortis par combinaison: ");

Console.WriteLine();

for (int i = 0; i < 49; i++)

{

int nb\_bon = combinaison\_gagnante[i];

if (nb\_bon == 0)

{

continue;

}

int numero = i + 1;

Console.WriteLine($"Le numéro {numero} est sorti {nb\_bon} fois.");

}

}

#endregion

#region 6 - CLASSE DES COMBINAISONS

/// <summary>

/// Procédure permettant de comparer les classes de combinaisons selon les tirages

/// </summary>

/// <param name="ratioCombinaison"></param>

/// <param name="tableau"></param>

/// <param name="win\_Lose"></param>

/// <param name="nbrCombinaisons"></param>

static void AfficherCombiConcordantes(string ratioCombinaison, int[,] tableau, int[] win\_Lose, int nbrCombinaisons)

{

if (win\_Lose.Length == 0)

{

return; //si il y a 0 gagnant, on écrit rien

}

Console.BackgroundColor = ConsoleColor.Blue;

Console.ForegroundColor = ConsoleColor.Black;

Console.Write($"{(double)win\_Lose.Length / (double)nbrCombinaisons \* 100.00}% des combinaisons {ratioCombinaison} sont gagnantes: ");

Console.WriteLine();

Console.ResetColor();

for (int i = 0; i < win\_Lose.Length; i++)

{

int winLose = win\_Lose[i];

Console.Write($"La combinaison #{winLose}: ");

for (int j = 0; j < 6; j++)

{

Console.Write(tableau[winLose, j].ToString().PadLeft(2, '0'));

Console.Write(" ");

}

Console.WriteLine();

}

Console.WriteLine();

}

#endregion

#region 7 - RECOMMENCER

/// <summary>

/// Fonction permettant de déterminer si l'utilisateur veut recommencer et mettre fin au programme s'il ne veut pas continuer.

/// </summary>

/// <returns>true lorsque l'utilisateur veut recommencer</returns>

public static bool Recommencer()

{

string str\_recommencer;

char recommencer;

// MESSAGE À L'UTILISATEUR S'IL VEUT RECOMMENCER LE PROCESSUS

do

{

Console.SetCursorPosition(0, 24);

Console.WriteLine(" ");

Console.SetCursorPosition(0, 24);

Console.ResetColor();

Console.Write("Voulez vous recommencer (O/N, puis appuyez sur ENTRÉE)? : ");

str\_recommencer = Console.ReadLine().ToUpper();

if (char.TryParse(str\_recommencer, out recommencer) == false || recommencer != 'O' && recommencer != 'N')

{

Console.BackgroundColor = ConsoleColor.Red;

Console.ForegroundColor = ConsoleColor.Black;

Console.SetCursorPosition(4, 23);

Console.WriteLine("Attention! Vous devez entrer O (Oui) ou N (Non) !\a");

Console.ResetColor();

}

} while (char.TryParse(str\_recommencer, out recommencer) == false || recommencer != 'O' && recommencer != 'N');

Console.Clear();

return recommencer == 'O';

}

#endregion

#region 8 - MESSAGE DE FIN DE PROGRAMME

/// <summary>

/// Procédure du message de remerciement et de fermeture à l'utilisateur

/// </summary>

public static void Message\_Fin\_Programme()

{

// MESSAGE D'APPUYER SUR ENTRÉE POUR QUITTER LE PROGRAMME

Console.WriteLine();

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("MERCI D'AVOIR UTILISÉ LE GÉNÉRATEUR DE NUMÉROS DE LOTERIE !");

Console.BackgroundColor = ConsoleColor.Black;

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Veuillez appuyer sur ENTRÉE pour quitter le programme.\a");

Console.ReadLine();

}

#endregion

}

}