// En MC :

// - Tableau reprenant les valeurs générées via la loi de Poisson

// - p(0) = 0,183

// - p(1) = 0,494

// - p(2) = 0,758

// - p(3) = 0,908

// - p(4) = 0,9716

// - p(5) = 0,99932

// - p(6) = 0,99932

// - p(7) = 1,00081

// couts présence dans le système

COUT\_1H\_ORDINAIRE = 25

COUT\_1H\_PRIORITAIRE = 40

// cout occupation dans station

COUT\_1H\_STATION\_PRIORITÈRE = 75

COUT\_1H\_STATION\_ORDINAIRE = 50

COUT\_1H\_STATION\_INOCCUPÉE = 20

COUT\_TRANSFORMATION\_CLIENT\_PRIO\_VERS\_ORDINAIRE = 30

MAX\_PRIORITAIRE\_FILE = 5

TAUX\_PRIORITAIRE = 0.3

// tableau en valeur cumulée

tabPoisson = [0.183, 0.494, 0.758, 0.908, 0.9716, 0.9932, 0.99932, 1.00081]

o───────────────────o ↓ nbStationsMin, nbStationsMax, tempsSimul

│ nbStationsOptimal │

o───────────────────o ↓ nbStationsOptimal

┌─── \*

│ nbStations = nbStationsMin

│╔══ while (nbStations ≤ nbStationsMax)

│║ file = 0

│║ filePrioritaire = 0

│║ fileCumulée = 0

│║ filePrioCumulée = 0

│║ nbTransfoClientTot = 0

│║ tempsInnocupéTot = 0

│║

│║ o──────────────o ↓ nbStations

│║ │ initStations │

│║ o──────────────o ↓ stations

│║ temps = 1

│║

│║╔══ while (temps ≤ tempsSimul)

│║║ // affichage des états de chaque stations pour les 20 premières minutes du nombre de station min

│║║┌── if( temps ≤ 20 AND nbStattions == nbStationsMin )

│║║│ sortir "AVANT PLACEMENT"

│║║│ i = 0

│║║│╔══ while ( i < nbStations )

│║║│║ sortir " - Station "+ i +" : "

│║║│║┌── if( station[i] ≠ 0)

│║║│║│ // ajouté le type de client

│║║│║│ sortir stations[i] +" clients."

│║║│║├── else

│║║│║│ sortir " pas de client."

│║║│║└──

│║║│║ i++

│║║│╙──

│║║│

│║║│ // affichage de la description de chaque file

│║║│ // infos nécessaire pour la compréhension des files ?

│║║│ sortir "File de client ordinaire : "+ fileOrdinaire +" client(s)."

│║║│ sortir "File de client prioritaire :"+ filePrioritaire +" client(s) (max : 5)"

│║║└──

│║║

│║║ o──────────────────o ↓ x0, a, c, m, tabLoiPoisson

│║║ │ nbArrivéesGénéré │

│║║ o──────────────────o ↓ nbArrivées, x0;

│║║ o────────────────────o ↓ nbArrivées, x0, a, c, m

│║║ │ répartitionArrivée │

│║║ o────────────────────o ↓ fileOrdinaire, filePrio, x0, nbTransformation

│║║ nbTransfoClientTot += nbTransformation

│║║ file += fileOrdinaire

│║║ filePrioritaire += filePrio

│║║

│║║┌── if(nbStattions == nbStationsMin AND temps ≤ 20 )

│║║│ sortir "Minutes "+ temps +" : "+ nbArrivées +" arrivées générées."

│║║│ // durée ??

│║║│ sortir fileOrdinaire +" clients ordinaires et "+ filePrio +" clients prioritaires."

│║║└──

│║║

│║║ o───────────────────o ↓ stations, x0, a, c, m, tempsInnocupé

│║║ │ repartiClientPrio │

│║║ o───────────────────o ↓ stations, filePrioritaires, x0, temspInnocupé

│║║

│║║ o───────────────o ↓ stations, x0, a, c, m, tempsInnocupé

│║║ │ repartiClient │

│║║ o───────────────o ↓ stations, file, filePrioritaires, x0, temspInnocupé

│║║

│║║ // additionne le nombre de client de chaque type pour avoir le temps d'attente de chaque type de client

│║║ filePrioCumulée += filePrioritaire

│║║ fileCumulée += file

│║║

│║║┌── if( nbStattions == nbStationsMin AND temps ≤ 20 )

│║║│ sortir "APRÈS PLACEMENT"

│║║│ i = 0

│║║│╔══ while ( i < nbStations )

│║║│║ sortir " - Station "+ i +" : "

│║║│║┌── if( station[i] ≠ 0)

│║║│║│ // ajouté le type de client

│║║│║│ sortir stations[i] +" clients."

│║║│║├── else

│║║│║│ sortir " pas de client."

│║║│║└──

│║║│║ i++

│║║│╙──

│║║│

│║║│ // affichage de la description de chaque file

│║║│ // infos nécessaire pour la compréhension des files ?

│║║│ sortir "File de client ordinaire : "+ fileOrdinaire +" client(s)."

│║║│ sortir "File de client prioritaire :"+ filePrioritaire +" client(s) (max : 5)"

│║║└──

│║║

│║║ temps ++

│║╙──

│║

│║ // différents coûts à calculer

│║ sortir "Coûts pour "+ nbStations +" stations."

│║ o─────────────o ↓ fileCumulée, filePrioCumulée, nbTransfoClientTot, tempsInnocupéTot

│║ │ afficheCout │

│║ o─────────────o ↓ total

│║

│║ // le nombre de station minimum n'est pas forcement 0

│║ // ⇒ on soustrais le nombre de station minimum du nombre de station pour avoir un indice de 0 à nbMaxStations - nbStations

│║ couts[nbStations - nbStationsMin] = total

│║

│║ nbStations ++

│║ o──────────────────o ↓ couts

│║ │ rechercheCoutMin │

│║ o──────────────────o ↓ nbStationsOptimal

│╙──

└──────────

o──────────────o ↓ nbStations

│ initStations │

o──────────────o ↓ stations

┌─── \*

│ i = 0

│╔══ while (i < nbStations)

│║ stations[i] = 0

│║ i++

│╙──

└──────────

o──────────────────o ↓ x0, a, c, m, tabLoiPoisson

│ nbArrivéesGénéré │

o──────────────────o ↓ nbArrivées, x0;

┌─── \*

│ o──────────o ↓ x0, a, c, m

│ │ génèreUN │

│ o──────────o ↓ un, x0

│

│┌── if (un < tabPoisson[0])

││ nbArrivées = 0

│├── else if (un < tabPoisson[1])

││ nbArrivées = 1

│├── else if (un < tabPoisson[2])

││ nbArrivées = 2

│├── else if (un < tabPoisson[3])

││ nbArrivées = 3

│├── else if (un < tabPoisson[4])

││ nbArrivées = 4

│├── else if (un < tabPoisson[5])

││ nbArrivées = 5

│├── else if (un < tabPoisson[6])

││ nbArrivées = 6

│├── else

││ nbArrivées = 7

│└──

└──────────

o──────────o ↓ x0, a, c, m

│ génèreUN │

o──────────o ↓ un, x0

┌─── \*

│ x1 = (a \* x0 + c) %m

│ un = x1/m

│ x0 = x1

└──────────

o────────────────────o ↓ nbArrivées, x0, a, c, m

│ répartitionArrivée │

o────────────────────o ↓ fileOrdinaire, filePrio, x0, nbTransformation

┌─── \*

│ fileOrdinaire = 0

│ filePrio = 0

│ nbTransformation = 0

│

│ i = 0

│╔══ while (i < nbArrivées )

│║ o──────────o ↓ x0, a, c, m

│║ │ génèreUN │

│║ o──────────o ↓ un, x0

│║┌── if (un < TAUX\_PRIORITAIRE )

│║│┌── if ( filePrio ≤ MAX\_PRIORITAIRE FILE )

│║││ filePrio++

│║│├── else

│║││ // gestion cout sup

│║││ nbTransformation++

│║││ fileOrdinaire++

│║│└──

│║├── else

│║│ fileOrdinaire++

│║└──

│║ i++

│╙──

└──────────

o───────────────────o ↓ stations, x0, a, c, m, tempsInnocupé

│ repartiClientPrio │

o───────────────────o ↓ stations, filePrioritaires, x0, temspInnocupé

┌─── \*

│ filePrioritaire = 0

│ // traitement du client prioritaire (stations num 1)

│ // regarde si la station est libre

│┌── if( stations[0] == 0 )

││ // si elle est libre, regarde si un client attend dans la file

││┌── if (filePrioritaire ≠ 0)

│││ // génère le temps pour le client, l'enlève de la file et l'ajoute dans la station

│││ filePrioritaire --

│││ o──────────────o ↓ x0, a, c, m, stations, 0

│││ │ duréeGénérée │

│││ o──────────────o ↓ stations, x0

│││ // enlève une minute de traitement

│││ stations[0] --

││├── else

│││ tempsInnocupéTot++

││└──

│├── else

││ stations[0]--

│└──

└──────────

o───────────────o ↓ stations, x0, a, c, m, tempsInnocupé

│ repartiClient │

o───────────────o ↓ stations, file, x0, temspInnocupé

┌─── \*

│ file = 0

│ // regarde chaque station pour

│ // - ajouter un client de la file si la station est libre

│ // - enlever une minute de traitement de chaque station

│ i = 1

│╔══ while (i ≤ nbStations)

│║ // regarde si la est station libre (temps d'attente de 0 )

│║┌── if (stations[i] == 0)

│║│ // regarde si un client est dans la file

│║│ // client dans la file prioritaire ont la priorité quand une station est vide

│║│┌── if( filePrioritaire ≠ 0)

│║││ filePrioritaire --

│║││ o──────────────o ↓ x0, a, c, m, stations, i

│║││ │ duréeGénérée │

│║││ o──────────────o ↓ stations, x0

│║││ stations[i] --

│║│├── else

│║││┌── if (file ≠ 0)

│║│││ file --

│║│││ o──────────────o ↓ x0, a, c, m, stations, i

│║│││ │ duréeGénérée │

│║│││ o──────────────o ↓ stations, x0

│║│││ stations[i] --

│║││├── else

│║│││ tempsInnocupéTot++

│║││└──

│║│├── else

│║││ stations[i] --

│║│└──

│║└──

│║ i++

│╙──

└──────────

o──────────────o ↓ x0, a, c, m, stations, ind

│ duréeGénérée │

o──────────────o ↓ stations, x0

┌─── \*

│ o──────────o ↓ x0, a, c, m

│ │ génèreUN │

│ o──────────o ↓ un, x0

│

│┌── if (un < 0.305)

││ T = 1

││┌── if (un < 0.661)

│││ T = 2

│││┌── if (un < 0.915)

││││ T = 3

││││┌── if (un < 0.966)

│││││ T = 4

│││││┌── if (un < 0.983)

││││││ T = 5

│││││├── else

││││││ T = 6

││││││

│││││└──

││││└──

│││└──

││└──

│└──

│ x0 = x1

│ stations[ind] = T

└──────────

o──────────────────o ↓ couts

│ rechercheCoutMin │

o──────────────────o ↓ nbStationsOptimal

┌─── \*

│ i = 1

│ coutMin = couts[0]

│ iStationOpti = 0

│╔══ while ( i < couts.length)

│║┌── if( couts[i] < coustMin)

│║│ coutMin = couts[i]

│║│ iStationOpti = i

│║└──

│║ i++

│╙──

│ nbStationsOptimal = iStationOpti + nbStationsMin

└──────────

o─────────────o ↓ fileCumulée, filePrioCumulée, nbTransfoClientTot

│ afficheCout │

o─────────────o

┌─── \*

│ total = 0

│ coutsPrésenceDansSysOrdinaire = fileCumulée / 60 \* COUT\_1H\_ORDINAIRE

│ sortir "cout de présence dans le système : "+ coutsPrésenceDansSysOrdinaire

│ total += coutsPrésenceDansSysOrdinaire

│

│ coutsPrésenceDansSysPrio = filePrioCumulée / 60 \* COUT\_1H\_PRIORITAIRE

│ sortir "cout de présence dans le système ( client prioritaire): "+ coutsPrésenceDansSysPrio

│ total += coutsPrésenceDansSysPrio

│

│ coutOccupationStationClassique = fileCumulée / 60 \* COUT\_1H\_STATION\_ORDINAIRE

│ sortir "cout d'occupation de station : "+ coutOccupationStationClassique

│ total += coutOccupationStationClassique

│

│ coutOccupationStationPrio = filePrioCumulée / 60 \* COUT\_1H\_STATION\_PRIORITÈRE

│ sortir "cout d'occupation de station ( client prioritaire ): "+ coutOccupationStationPrio

│ total += coutOccupationStationPrio

│

│ coutTransformationClient = nbTransfoClientTot \* COUT\_TRANSFORMATION\_CLIENT\_PRIO\_VERS\_ORDINAIRE

│ sortir "cout de transformation de client classique vers prioritaire : "+ coutTransformationClient

│ total += coutTransformationClient

│

│ coutInnocupation = tempsInnocupéTot / 60 \* COUT\_1H\_STATION\_INOCCUPÉE

│ sortir "cout d'innocupation d'une station : "+ coutInnocupation

│ total += coutInnocupation

└──────────