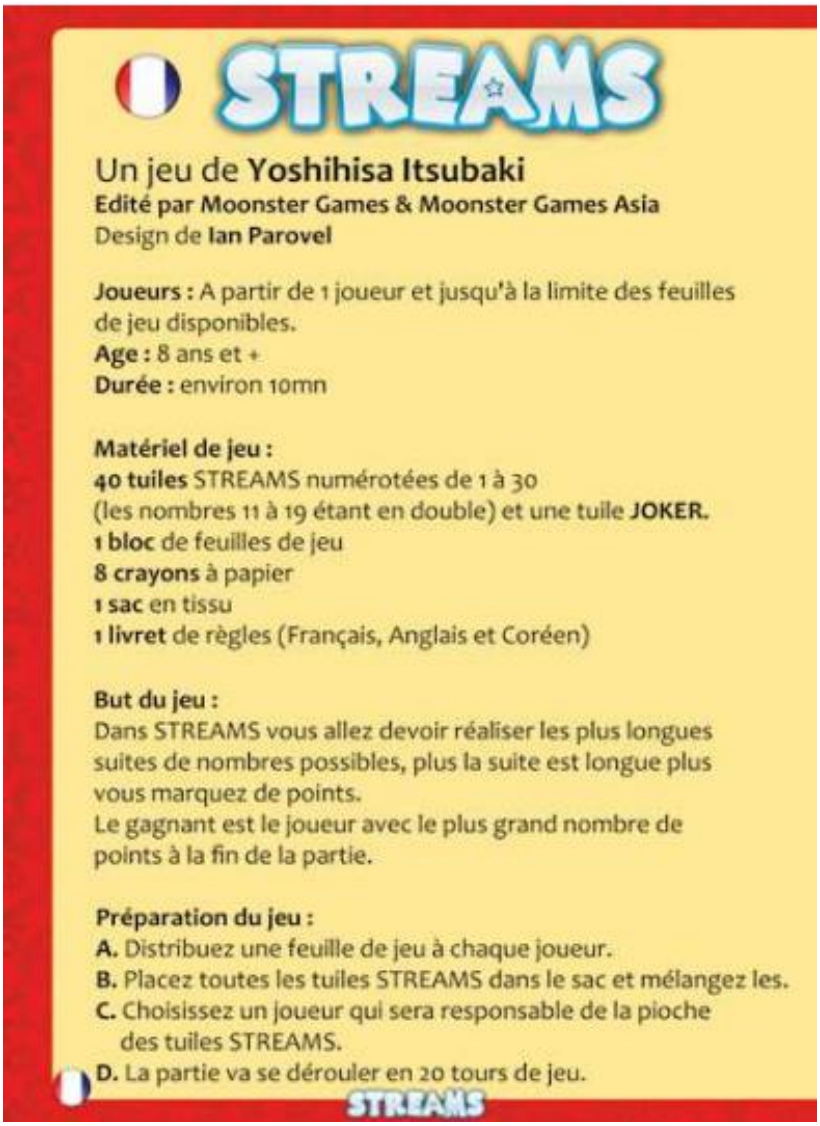
BIN2181 – LAS

HE Vinci 2023-2024

Projet de programmation système : Streams

Le projet à concevoir est la simulation du jeu "Streams".

1. Le jeu



STREAMS

Un jeu de **Yoshihisa Itsubaki**
Edité par Moonster Games & Moonster Games Asia
Design de **Ian Parovel**

Joueurs : A partir de 1 joueur et jusqu'à la limite des feuilles de jeu disponibles.
Age : 8 ans et +
Durée : environ 10mn

Matériel de jeu :
40 tuiles STREAMS numérotées de 1 à 30 (les nombres 11 à 19 étant en double) et une tuile **JOKER.**
1 bloc de feuilles de jeu
8 crayons à papier
1 sac en tissu
1 livret de règles (Français, Anglais et Coréen)

But du jeu :
Dans STREAMS vous allez devoir réaliser les plus longues suites de nombres possibles, plus la suite est longue plus vous marquez de points.
Le gagnant est le joueur avec le plus grand nombre de points à la fin de la partie.

Préparation du jeu :
A. Distribuez une feuille de jeu à chaque joueur.
B. Placez toutes les tuiles STREAMS dans le sac et mélangez les.
C. Choisissez un joueur qui sera responsable de la pioche des tuiles STREAMS.
D. La partie va se dérouler en 20 tours de jeu.

STREAMS

Déroulement d'un tour de jeu :

E. Le joueur responsable de la pioche tire au hasard une tuile STREAMS dans le sac, il annonce à voix haute le nombre indiqué dessus ou s'il s'agit de la tuile JOKER, puis place celle-ci face visible sur la table.

F. Tous les joueurs inscrivent le nombre annoncé (ou le symbole JOKER) sur leurs feuilles de jeu.
Ils sont libres de placer le nombre ou le JOKER où ils veulent dans une case libre de la grille, ils essaient ainsi de constituer progressivement la ou les suites de nombres les plus longues possibles.

G. Dès que tous les joueurs ont inscrit le nombre sur leurs feuilles de jeu, un nouveau tour commence (reprendre au point **E**).

Fin de Partie :

H. A la fin du 20ème tour, lorsque la grille de chaque joueur est remplie, la partie s'arrête, les joueurs comptent leurs scores (voir page 2) en se reportant au tableau des scores inclus sur leurs feuilles de jeu.

I. La partie est remportée par le(s) joueur(s) avec le plus haut score.

Dans STREAMS, qu'est ce qu'une suite ?
Une suite est une séquence de nombres allant du plus faible au plus élevé, ceci incluant :

- Les nombres qui ne sont pas consécutifs mais dont les valeurs vont dans l'ordre croissant.
Par exemple, 4-6-17-23-25-29 est une suite de six nombres.

- Les nombres identiques placés à la suite les uns des autres.
Par exemple, 7-10-11-11-15 constitue une suite de cinq nombres.

- Le joker, qui doit être comptabilisé dans une seule suite et qui doit respecter les deux critères précédents.
Par exemple, 5-12-X-14-17-21-21-27 est une suite de huit nombres valide.

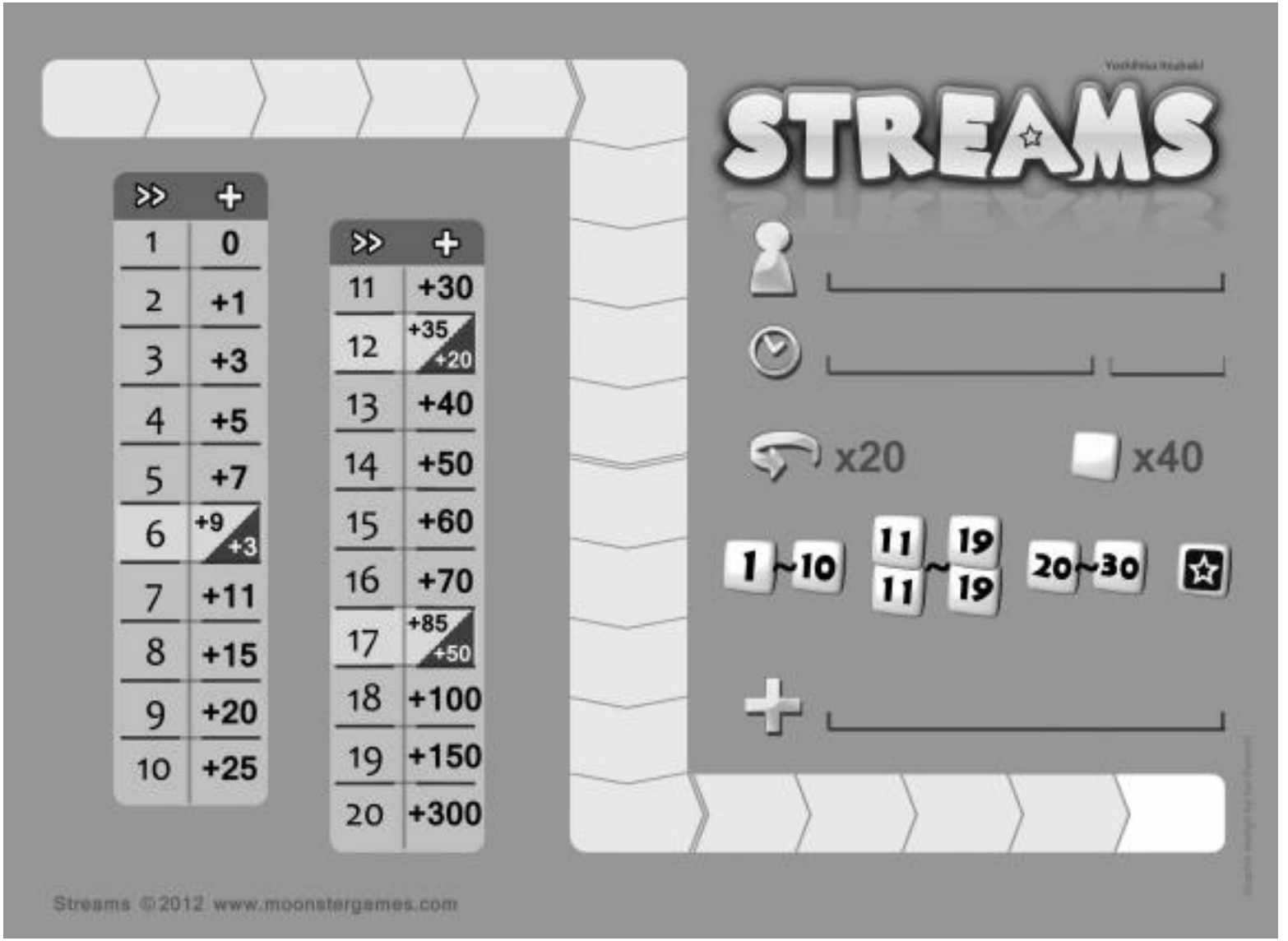
Décompte des points (en mode normal) :
Une fois la partie terminée, les joueurs doivent identifier chaque suite d'au moins deux nombres figurant sur leur feuille.

Chaque suite rapporte un nombre de points variant selon sa longueur. Pour les suites de 6, 12 et 17 nombres, prendre la valeur de gauche dans le tableau de référence.

Le score final d'un joueur est le total de points rapporté par chacune des suites qu'il a pu constituer sur sa fiche.

Décompte des points (en mode expert) :
Faire comme dans le décompte des points (en mode normal), sauf pour les suites de 6,12 et 17 nombres, qui rapportent le nombre de points indiqué à droite dans le tableau de référence. En mode expert, ces suites rapportent moins de points et il faudra donc les éviter.

STREAMS


STREAMS
» +
1 0
2 +1
3 +3
4 +5
5 +7
6 +9 +3
7 +11
8 +15
9 +20
10 +25
» +
11 +30
12 +35 +20
13 +40
14 +50
15 +60
16 +70
17 +85 +50
18 +100
19 +150
20 +300
x20
x40
1~10
11~19
11~19
20~30
Streams ©2012 www.moonstergames.com

2. Implémentation du jeu

Le projet à réaliser est une application client-serveur qui utilise des sockets TCP/IP pour implémenter le dialogue entre le gestionnaire du jeu (le serveur) et les joueurs (les clients) et une mémoire commune pour partager les scores des joueurs.

Un joueur (un client) demande l'autorisation de jouer au gestionnaire du jeu (le serveur). Pour ce faire, il communique son nom au serveur. Si la partie n'a pas encore démarré et si le nombre maximum de joueurs n'est pas atteint, le joueur est accepté, sinon il est refusé. Le serveur envoie sa réponse au joueur.

Avant de démarrer une partie le serveur attend que les deux conditions suivantes soient remplies :

1. Il y a au moins deux joueurs.
2. Les joueurs ont été acceptés durant une phase d'inscription, qui dure 30 secondes (cette condition permet d'accepter plus deux joueurs en évitant de démarrer une partie dès que deux joueurs sont inscrits).

Le serveur crée les tuiles c'est-à-dire 40 pièces numérotées de 1 à 30 (de 11 à 19 les pièces sont dédoublées) et un joker.

Une partie comprend 20 tours de jeu.

Un tour de jeu se déroule comme suit :

- le serveur tire une tuile au hasard et la communique à tous les joueurs.
- chaque joueur affiche la tuile envoyée et demande à l'utilisateur à quelle position, dans la grille, il souhaite la mettre. Si la position est occupée, la tuile sera placée à droite de la position demandée. Dès que la tuile a reçu une place, le joueur le signale au serveur et affiche la grille mise à jour.
- le tour se termine quand le serveur a été prévenu que tous les joueurs ont placé la tuile.

Après 20 tours de jeu, chaque joueur calcule son score. Pour calculer ses points, le client commence par détecter les suites de nombres dans la grille. Une suite se termine dès qu'un nombre est strictement plus petit que celui qui le précède. Chaque suite rapporte des points en fonction de sa longueur, conformément au tableau suivant :

>>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
+	0	1	3	5	7	9	11	15	20	25	30	35	40	50	60	70	85	100	150	300

Une fois ses points calculés, le client les envoie au serveur.

Pratiquement, les informations circulent du serveur vers les joueurs et des joueurs vers le serveur via les sockets. Les noms des joueurs et leurs scores sont conservés en mémoire partagée.

Une fois que tous les joueurs ont envoyé leurs points, le serveur génère le *ranking* en triant la mémoire partagée par ordre décroissant des points. Pour finir, le serveur publie le ranking (càd les noms des joueurs triés selon leurs scores) qui sera affiché par chaque joueur.

Exemple de déroulement d'une partie

Initialisation de la partie

Le serveur initialise un timer de 30 secondes pour commencer la phase d'inscription.
Alice envoie une requête d'inscription au serveur.
Le serveur accepte l'inscription.
Bob envoie une requête d'inscription au serveur.
Le serveur accepte aussi l'inscription de Bob.
Au bout de 30 secondes, la phase d'inscription est terminée.
Carl envoie une requête d'inscription au serveur. Sa demande est laissée en attente.

Initialisation de la phase de Jeu

Au terme des 30 secondes de la phase d'inscription, le serveur crée les tuiles numérotées de 1 à 30 (avec doublons pour 11-19) et un joker.

Premier tour

Le serveur tire une tuile au hasard, disons le numéro 5, et l'envoie à Alice et Bob. (Le serveur se place en attente d'un message indiquant qu'Alice et de Bob ont joué.)
Alice reçoit la tuile 5, décide de la placer en position 2 sur sa grille et signale au serveur qu'elle a joué.
Bob reçoit également la tuile 5 et choisit de la placer en position 1, et indique au serveur qu'il a placé sa tuile.
La réception par le serveur des messages de Alice et de Bob clôture le tour.
Les ordinateurs d'Alice et Bob affichent chacun la grille de jeu du joueur mise à jour pour refléter les choix opérés.
Note: Il est tout-à-fait possible que Bob réponde au server avant qu'Alice n'ait pris sa décision.

Tours 2 à 20

Les tours 2 à 20 se déroulent exactement de la même façon que le tour 1. Mais à chaque tour, la tuile tirée par le serveur et les décisions prises par Alice et Bob peuvent être différentes (évidemment !).

Fin de la Partie

Alice et Bob analysent chacun leur grille de jeu pour détecter les suites de nombres. Ils utilisent le tableau de calcul des points pour connaitre leur score puis envoient celui-ci au serveur.
Supposons qu'Alice ait formé une suite de 4 tuiles consécutives, une de 7 tuiles consécutive et une de 2. Selon le tableau de scores, elle marque donc 5 pour la suite de 4 tuiles, 9 pour la suite de 7 tuiles et 1 point pour la suite de 2. Elle obtient donc un score total de 15 points.
Bob a su faire deux suites de tuiles: l'une de 12 tuiles successives et une de 4 tuiles. Il obtient donc un score de 35 points pour la suite de 12 tuiles et 5 points pour la suite de 4 tuiles. Il obtient donc un total de 40 points.
Alice envoie un message au serveur pour lui indiquer qu'elle a obtenu 15 points.
Bob envoie un message au serveur pour lui indiquer qu'il a obtenu 40 points.
Le serveur trie les joueurs selon leurs scores dans la mémoire partagée. Bob arrive donc premier et Alice seconde.
Alice et Bob consultent le classement en mémoire partagée et l'affichent.
Le serveur relance une phase d'inscription.
Restée en attente, la demande d'inscription de Carl est traitée.

3. Indications de programmation

Il y a 2 programmes à écrire : le serveur et le joueur, ce dernier est le client.

Pour l'implémentation des sockets, le numéro du port est à préciser comme argument des programmes.

Le serveur ne s'éteint jamais. Cependant, dans le cadre de ce projet, il doit être possible de l'arrêter en lui envoyant un signal SIGINT (Ctrl-C). Notez que le serveur ne peut pas interrompre la partie en cours. Cela signifie que la phase d'inscription de 30 secondes et la phase de jeu ne seront pas interrompues par Ctrl-C. A la réception d'un SIGINT, le serveur s'arrêta après la fin de la partie courante.

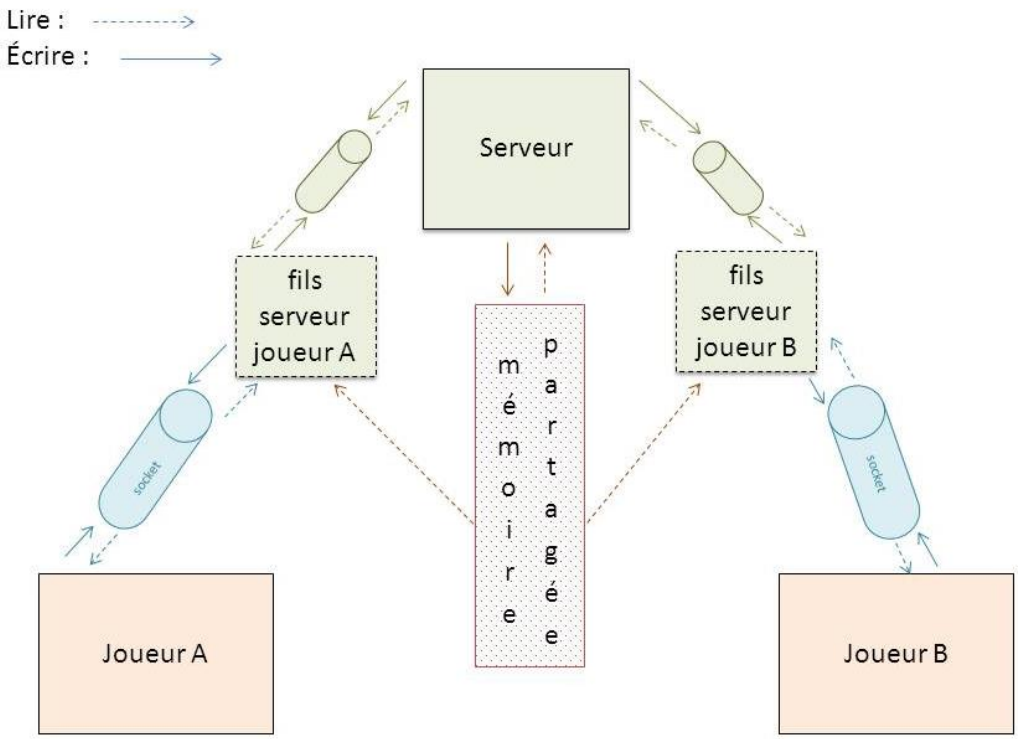
L'accès à la mémoire partagée doit être sécurisé par l'utilisation de sémaphores.

Le serveur est responsable de la création et de la destruction des IPC.

4. Utilisation des moyens de communications

Les clients et le serveur ne tournent pas nécessairement sur la même machine. La mémoire partagée n'est accessible que par des processus qui tournent sur la même machine que le serveur. Pour ce faire, le serveur doit créer un fils par client qui joue le rôle de client local et accède à la mémoire partagée. Les clients distants communiquent alors avec le fils qui leur est attitré (via sockets) et effectue les affichages et autres entrées/sorties à destination du joueur. Le serveur communique avec ses fils via des pipes.

Les informations qui circulent entre un joueur et le serveur transitent par des "sockets" TCP et les informations connues de tous sont conservés dans une "mémoire partagée".



Les sémaphores permettront aux clients locaux (fils du serveur) de se mettre en attente du ranking généré par le serveur : dès qu'un client local a relayé au serveur le score communiqué par un joueur, il demande l'accès à la mémoire partagée pour pouvoir lire le ranking et l'envoyer au joueur (client).

5. Contraintes pratiques

Pour mener à bien ce projet, il est indispensable de travailler de manière modulaire ; il vous est demandé de respecter la découpe suivante :

- network (tcp-ip)
- ipc
- jeu
- serveur (*main*)
- client (*main*)

Pour compiler et faire l'édition de liens des programmes, un « makefile » doit être écrit.

Quand l'application se termine, aucun IPC ne peut traîner sur le système ! ! !

Tous les appels systèmes doivent être testés. En cas d'erreur, vous pouvez simplement fermer l'application.

Les entrées/sorties (clavier/écran) ne doivent pas être réalisés à l'aide de syscalls. Vous pouvez utiliser la fonction *printf* ainsi que les fonctions de lecture du module *utils*.

Nous vous conseillons d'utiliser le modules *utils* qui vous a été fourni pour cette UE.

Il ne faut pas prévoir le cas où l'un ou l'autre processus serait tué pendant une partie. Vous pouvez également supposer que le joueur ne se trompera jamais en entrant des données.

Enfin, pour faciliter les tests de votre application, le programme serveur doit pouvoir être lancé avec un fichier en second argument de la ligne de commande. Lorsqu'un tel fichier est fourni au serveur, ce dernier ne tirera pas les tuiles au hasard mais lira les valeurs des tuiles séquentiellement dans le fichier (à raison d'une valeur par ligne, en codant le joker avec la valeur 31) [1]. Vous ne devez pas utiliser les syscalls pour traiter ce fichier de test.

6. Délivrables

6.1. L'analyse

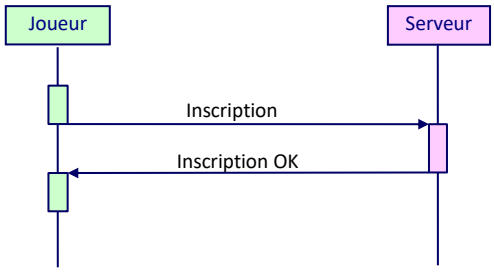
Nous vous demandons de réaliser l'analyse de l'application.

Avant de se lancer dans la programmation, il est en effet prudent de commencer par analyser le problème posé. Il vous est demandé d'établir une liste des messages qui vont être échangés entre le serveur et les joueurs, d'en fixer le format et de concevoir les scénarios. Par exemple,

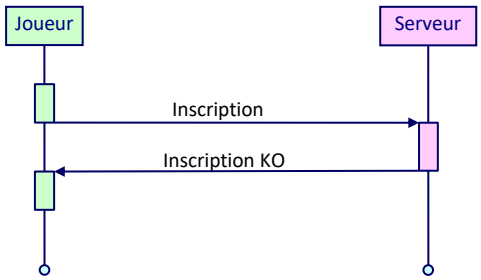
[1] On supposera qu'il y a suffisamment de suites de 20 tuiles dans le fichier de test pour pouvoir jouer plusieures parties d'affilée.

pour l'inscription d'un joueur, le dialogue entre le serveur et le joueur peut se représenter comme suit :

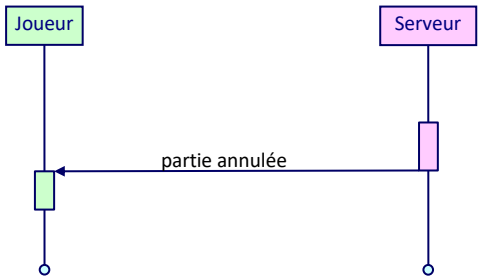
- Inscription acceptée :



- Inscription refusée :



- Annulation de la partie (s'il n'y a pas assez de joueurs inscrits) :



Outre la communication, il faut réfléchir à la découpe de votre application. Au terme de cette analyse, nous vous demandons le Makefile ainsi qu'un document indiquant la découpe en fonctions de votre solution. Voir fichier canevas « analyse_application.docx ».

La validation de vos protocoles de communication, de votre Makefile et de la découpe en fonctions sera faite en séance durant la première semaine (voir Planning ci-dessous).

6.2. Le code final

Nous vous demandons de remettre sur MooVin l'ensemble de votre code source (documenté!), ainsi qu'un Makefile permettant la compilation des différents programmes exécutables au terme des deux semaines du projet (voir Planning ci-dessous).

Nous vous rappelons que la politique de la Haute Ecole en matière de plagiat est très stricte. Nous vous signalons qu'un système automatique de détection de plagiat sera appliqué à votre code.

6.3. Défense orale

Vos projets seront testés en votre présence le **jeudi 16 mai**. Ces tests seront effectués sur les machines de l'école OU vos machines au choix. Nous vous conseillons cependant plutôt l'emploi de vos machines pour éviter des soucis de comptabilité. Le code qui sera exécuté sera celui soumis sur MooVin et pas un autre ! Nous vous le refournirons pour les tests. Le scénario de test vous sera communiqué durant le projet.

7. Planning

Date	Action
8-12 avril 2024 à 14h	Choix des groupes de 3 étudiants (sur Moovin)
Jeudi 11 avril 2024 à 16h	Présentation du projet (Auditoire A)
Lundi 15 avril 2024	1e séance du projet
Vendredi 19 avril 2024	Analyse validée en séance durant la 1e semaine
Mardi 30 avril 2024 à 20h	Remise du code source de vos programmes
Jeudi 16 mai 2024	Défense orale du projet (un horaire précis + scénario de tests vous sera communiqué en temps voulu)

8. Evaluation

La note du projet se base sur l'évaluation de votre code et la défense du projet (qualité du code + exécution des tests lors de la défense).

Pour rappel, le projet compte pour 25% de la note d'UE de juin.

Bon travail !