# BREAK THE CODE

## #BrestEdition2024

L'Objectif du Break The Code (BTC) est de récupérer un maximum de points dans le temps imparti (1h30) en équipe de 2 à 3 personnes.

Les exercices sont regroupés dans ce fichier PDF et les inputs associés à chaque exercice seront mis à disposition dans un dossier éponyme.

Chaque exercice rapporte un nombre de points en fonction de sa difficulté. Résoudre en premier un exercice, donne droit à des points bonus. Pour soumettre un exercice, il suffit d'utiliser la plateforme CTFd mise à disposition. Il est indispensable de respecter la mise en forme de la réponse comme indiqué dans le sujet, sinon elle sera rejetée !

Un scoreboard est disponible sur la plateforme. À la fin du temps imparti, le scoreboard est gelé pendant 10 min mais les réponses seront toujours prises en compte.

En plus des exercices classiques, un exercice d'optimisation est disponible. Le classement d'optimisation se fera en fonction du score de la solution, en cas d'égalité, l'ordre de remise est pris en compte (la première équipe à avoir rendu sa solution).

## Tableau des points par exercice :

| Exercice | Total de points | Points Bonus |
|---|---|---|
| Tilted Loot | 21 | 3 |
| Storm Survival | 17 | 8 |
| Here Comes A New Challenger! | 58 | 4 |
| Bang ! | 58 | 4 |
| Victory Royale | 37 | 7 |
| Target Acquired | 47 | 4 |
| Fortnite Championship Series | 42 | 4 |
| Team Rebuild | 50 | 4 |
| Sniper Forever | 53 | 4 |
| Is This A Fair Tournament ? | 27 | 3 |

## Tableau des points de l'exercice d'optimisation :

| Classement | Nombre de points |
|---|---|
| 1 | 55 |
| 2 | 41 |
| 3 | 34 |
| 4 | 28 |
| 5 | 23 |
| 6 à 10 | 18 |
| 11 à 15 | 12 |
| 16 et plus | 5 |

# 1 - Tilted Loot

Dans un Battle Royal, il est très important de commencer avec un bon stuff pour pouvoir rouler sur les autres.

Pour cet exercice, il faudra déterminer la meilleure zone de drop pour s'assurer la victoire.

La carte se présente sous la forme d'un ensemble de cases, qui contiennent un nombre d'objets. La carte est un carré de 100 cases de côté, et chaque case contient de 0 à 100 objets.

Il faudra trouver la case qui est au centre d'une croix pour laquelle la somme des valeurs contenues dans cette croix est la plus élevée de la carte.

Précision :

- La numérotation commence à 0

**Input :** Le fichier ***carte_brest.csv*** qui contient les informations ci-dessous.

**N** lignes avec les valeurs de chaque case séparée par des virgules.

**Output :**
Les coordonnées du centre de la croix dans la forme suivante :
coordonnéeX;coordonnéeY;totalDeLaCroix avec :

- coordonnéeX : la colonne de la case au centre de la croix
- coordonnéeY : le numéro de ligne.
- totalDeLaCroix : somme des valeurs de la croix

**Exemple :**

Input :

*68,54,57,75,65,75*

*77,54,30,27,74,84*

*72,10,55,95,3,43*

*42,74,23,66,81,9*

*0,77,97,89,80,33*

*47,53,11,9,10,4*

La case rouge est la case sur laquelle il faut atterrir, les cases vertes sont adjacentes et serviront à récupérer d'autres objets.

| | | | | | |
|---|---|---|---|---|---|
| 68 | 54 | 57 | 75 | 65 | 75 |
| 77 | 54 | 30 | 27 | 74 | 84 |
| 72 | 10 | 55 | 95 | 3 | 43 |
| 42 | 74 | 23 | 66 | 81 | 9 |
| 0 | 77 | 97 | 89 | 80 | 33 |
| 47 | 53 | 11 | 9 | 10 | 4 |

Output : 3;3;354

# 2 - Storm Survival

Lorsque vous jouez à ***Fortnite***, diverses mécaniques de jeu rentrent en compte pour rendre votre victoire plus ou moins (in)certaine.

Au-delà de la compétition entre les différents joueurs de la **Battle Royale**, la tempête est un phénomène qui intervient de manière certaine sur toutes les parties.

Dans certaines parties vous vous retrouvez proches de la tempête parce que vous avez souhaité récupérer un **loot** excentré sur la carte, dans d'autres parties vous êtes plus proches de la zone de sûreté car vous avez géré votre temps et position stratégiquement.

À partir de différentes positions et vitesses de votre joueur et de la tempête durant les parties, vous devez déterminer le nombre de fois où vous êtes certain d'arriver à la zone de sûreté avant que la tempête ne vous atteigne.

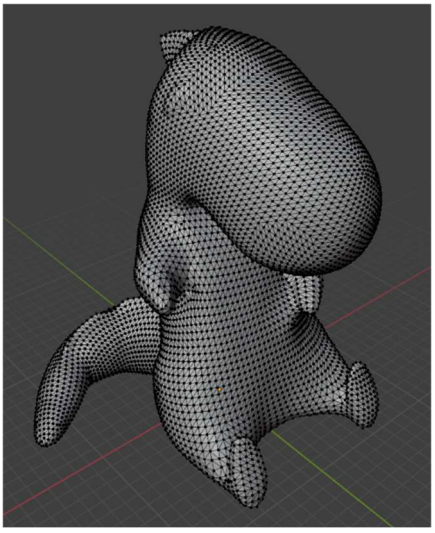
Considérez que vous et la tempête évoluez dans un environnement en une dimension (1D).

**Input :** Fichier ***storm_survival_brest.csv*** au format suivant *player_safe_distance,player_speed,storm_player_distance,storm_speed* avec :

- *player_safe_distance* : la distance entre vous et la zone de sûreté (si négative, alors vous êtes déjà dans la zone de sûreté)
- *player_speed* : votre vitesse de course, en se dirigeant vers la zone de sureté
- *storm_player_distance* : la distance entre la tempête et vous (si négative, alors la tempête vous a déjà atteint, trop tard !)
- *storm_speed* : la vitesse de la tempête, se dirigeant vers la zone de sureté

**Output :** Le nombre de fois où vous êtes arrivé dans la zone de sûreté avant que la tempête ne vous atteigne.

# 3 – Here Comes a New Challenger !



Notre mascotte TechRex aimerait jouer elle aussi ! Mais elle est exigeante et nous la gâtons trop, donc nous lui avons confectionné un skin avec sa propre apparence. Il s'agit d'un modèle 3D, autrement dit un ensemble de faces mises bout à bout dont les sommets ont chacun 3 coordonnées (X,Y,Z).

Quand nous lui avons montré le résultat, à notre grand désarroi, notre mascotte afficha cependant un air sceptique. « J'aime ce côté mignon que vous m'avez donné, mais je voudrais être impressionnant aussi. Que je fasse trembler les verres d'eau quand je marche ! »

Nous allons devoir déterminer à quel point notre petit dino est imposant, afin de pouvoir y remédier. Nous cherchons alors (de manière assez arbitraire) la surface totale prise par le skin.

Vous devrez déterminer la somme totale des aires de chaque face du modèle 3d.

**Input :** Le fichier ***techrex.obj*** qui contient le modèle 3D au format usuel OBJ de Wavefront.

Précisions :

- Toutes les faces sont **triangulaires, quelconques et planes**.
- Les sommets sont toujours référencés par leur index (commençant à l'index 1), suivant l'ordre de leur déclaration.

**Output :** La somme totale des aires de chaque face.

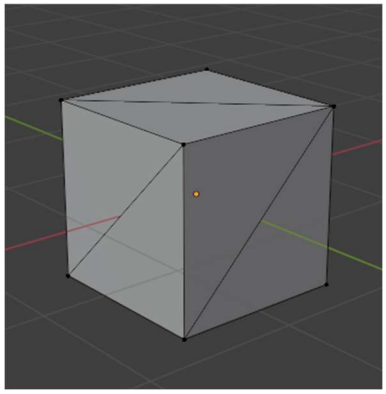
Le résultat doit être **arrondi à l'entier le plus proche**.

**Exemple :**

Un cube de 1.4 unités de côté est composé de 8 sommets et 6 faces carrées, donc de 12 faces triangulaires. Chaque face triangulaire a une aire de 0.979999… donc le cube a une surface totale de 11.759999… unités, ce qui donne un résultat de 12 unités une fois arrondi à l'entier le plus proche.

Input (simplifié) :

```
v 0.700000 0.700000 -0.700000
v 0.700000 -0.700000 -0.700000
v 0.700000 0.700000 0.700000
v 0.700000 -0.700000 0.700000
v -0.700000 0.700000 -0.700000
v -0.700000 -0.700000 -0.700000
v -0.700000 0.700000 0.700000
v -0.700000 -0.700000 0.700000
vn -0.0000 1.0000 -0.0000
vn -0.0000 -0.0000 1.0000
```



```
vn -1.0000 -0.0000 -0.0000
vn -0.0000 -1.0000 -0.0000
vn 1.0000 -0.0000 -0.0000
vn -0.0000 -0.0000 -1.0000
vt 0.875000 0.500000
vt 0.625000 0.750000
vt 0.625000 0.500000
vt 0.375000 1.000000
vt 0.375000 0.750000
vt 0.625000 0.000000
vt 0.375000 0.250000
vt 0.375000 0.000000
vt 0.375000 0.500000
vt 0.125000 0.750000
vt 0.125000 0.500000
vt 0.625000 0.250000
vt 0.875000 0.750000
vt 0.625000 1.000000
f 5/1/1 3/2/1 1/3/1
f 3/2/2 8/4/2 4/5/2
f 7/6/3 6/7/3 8/8/3
f 2/9/4 8/10/4 6/11/4
f 1/3/5 4/5/5 2/9/5
f 5/12/6 2/9/6 6/7/6
f 5/1/1 7/13/1 3/2/1
f 3/2/2 7/14/2 8/4/2
f 7/6/3 5/12/3 6/7/3
f 2/9/4 4/5/4 8/10/4
f 1/3/5 3/2/5 4/5/5
f 5/12/6 1/3/6 2/9/6
```

# 4 – Bang !

Quand on se fait tirer dessus, c'est important de savoir d'où provient le coup de feu.

Avec votre team, vous allez pouvoir vous entraider pour savoir quel est le petit malin qui a essayé de vous tirer dessus.

Pour ce faire, il vous faudra utiliser les fichiers mp3 qui contiennent les enregistrements de ce coup de feu, pris à 3 endroits différents. Les enregistrements sont synchronisés et démarrent au moment où le coup de feu est parti.

A partir de ces enregistrements, il vous faudra déterminer où il était pour aller l'éliminer et faire un top 1.

Vous pouvez considérer que les conditions climatiques sont normales, avec une vitesse du son de 343m/s.

Tous les joueurs se déplacent dans un référentiel de 1000m de côté, sans changement d'altitude.

L'atténuation du bruit n'est pas à prendre en compte.

Voici les coordonnées où se trouvent les joueurs :

| Joueur | X | Y |
|---|---|---|
| 1 | 870 | 648 |
| 2 | 38 | 912 |
| 3 | 115 | 186 |

**Input :** Un fichier audio par joueur contenant ce qu'il entend.

**Output :** Les coordonnées du tireur au format X;Y avec :

- X : l'abscisse du tireur
- Y : l'ordonnée du tireur

Elles doivent être arrondies à l'entier le plus proche.

**Exemple :**

Input :

Fichier audio avec un coup de feu débutant à 0,291s

Position du joueur : 0;100

Le tireur est donc situé en coordonnée 0;0

Output : 0;0

# 5 – Victory Royale

Afin de faire son premier TOP 1, notre joueur souhaite connaître quels sont ses points forts afin de s'améliorer. Pour cela, nous avons accès à deux fichiers répertoriant :

- Les caractéristiques de chaque arme utilisée
- Les duels qu'il a fait contre d'autres joueurs.

Le fichier des duels contient 1000 duels totalement indépendants les uns des autres. Pour chaque duel, les joueurs sont à une certaine distance (courte, moyenne ou longue) et possèdent chacun une seule arme (pompe, fusil d'assaut ou sniper).

À chaque fois, le joueur et son adversaire auront 200 points de vie. Pour gagner un combat, il faut que l'adversaire n'ait plus de points de vie avant que notre joueur perde tous ses points de vie. Si les deux joueurs perdent leur point de vie en même temps on considère que c'est un match nul et compte comme 0,5 victoires, une «demi»-victoire. Il faut prendre en compte les caractéristiques des armes utilisées ainsi que la distance de chaque combat. La précision agit comme un multiplicateur de dégâts selon la distance.

Vous devez répondre à ces deux questions que notre joueur se pose :

1. Quel est le pourcentage de chance qu'il tue un adversaire en ayant un pompe ?
2. Quel est le pourcentage de chance qu'il tue un adversaire en étant à longue distance ?

**Input :**

1. Fichier armes.csv : Ce fichier contient les caractéristiques des différentes armes disponibles. Chaque ligne représente une arme avec les colonnes suivantes :
   - arme : Nom de l'arme.
   - degat : Dégâts infligés par l'arme.
   - precision_courte : Précision de l'arme à courte distance.
   - precision_moyenne : Précision de l'arme à moyenne distance.
   - precision_longue : Précision de l'arme à longue distance.
   - cadence : Nombre de tirs par seconde.

2. Fichier duels_Brest.csv : Ce fichier contient la liste des duels à simuler. Chaque ligne représente un duel avec les colonnes suivantes :
   - joueur_arme : Nom de l'arme utilisée par le joueur.
   - adversaire_arme : Nom de l'arme utilisée par l'adversaire.
   - distance : Distance du duel (courte, moyenne, ou longue).

**Output**

Réponse sous la forme «A B» avec :

- A : réponse à la question 1, un nombre arrondi à 2 chiffres après la virgule.
- B : réponse à la question 2, un nombre arrondi à 2 chiffres après la virgule.

**Exemple**

Input :

*arme,degat,precision_courte,precision_moyenne,precision_longue,cadence*
*FusilAssaut,36,0.35,0.30,0.15,5.5*
*Pompe,88,3,0.2,0.01,1.5*
*Sniper,132,0.40,1,2,0.33*

*Sniper,FusilAssaut,longue*

*Pompe,Sniper,longue*

*FusilAssaut,Pompe,courte*

*Pompe,Pompe,courte*

*FusilAssaut,Pompe,longue*

Output : 25,00 66,67

# 6 – Target Acquired

Ça y est, vous êtes parachutés dans une nouvelle partie ! Dans ce mode de jeu, c'est du chacun pour soi : vous êtes seul contre 98 adversaires. Une fois au sol, les joueurs se dispersent, ramassent des armes à distance, construisent et se cachent.

De là où vous êtes, vous avez une assez bonne vision de toute la carte et vous pouvez voir certains de vos adversaires. Vous ne les voyez pas tous cependant : certains sont à l'abri derrière des murs. Vous vous demandez alors :

Si vous deviez tirer depuis votre position, combien de joueurs auriez-vous la possibilité de toucher avec un headshot (tir à la tête) ?

Précisions :

- Nous ignorons la hauteur en considérant une carte 2D horizontale, sous forme d'une grille de cases carrées de même taille. Chaque case peut contenir un mur, un adversaire, ou vous-même (bien entendu, vous n'êtes que dans une seule case).
- Les murs et les joueurs ont tous une hitbox (boîte de collision) qui s'étend dans toute la case sur laquelle ils sont placés. Personne ne bouge, ni vous ni vos adversaires.
- La tête d'un joueur se trouve au centre de sa case. Vos tirs partent du centre de votre case. Ainsi, **on considère le segment reliant le centre du joueur au centre d'un adversaire. Ce segment ne doit être interrompu par aucun mur.** Attention, un joueur peut en cacher un autre : votre vision (i.e. ce segment) ne peut être bloquée que par des murs, pas par d'autres adversaires.
- Vous pouvez voir plusieurs adversaires, mais il vous faudrait plusieurs tirs dans des directions différentes : il faut compter tous ces adversaires dans le résultat. C'est bien le nombre d'adversaires (à toucher en leur centre) qui doit être comptabilisé.

**Input :** Le fichier ***target_acquired.txt*** qui contient les informations ci-dessous.

Une première ligne avec la valeur N : un entier indiquant la taille NxN de la carte, et donc le nombre de caractères N d'une ligne de la grille (sans compter le saut de ligne à la fin).

Les N lignes suivantes (contenant N caractères + le saut de ligne) composent la grille NxN qui représente la carte. Voici la signification de chaque caractère :

- F « floor » : la case est vide
- P « player » : c'est votre position
- E « enemy » : un adversaire se trouve sur cette case
- W « wall » : la case est un mur et bloque votre vision

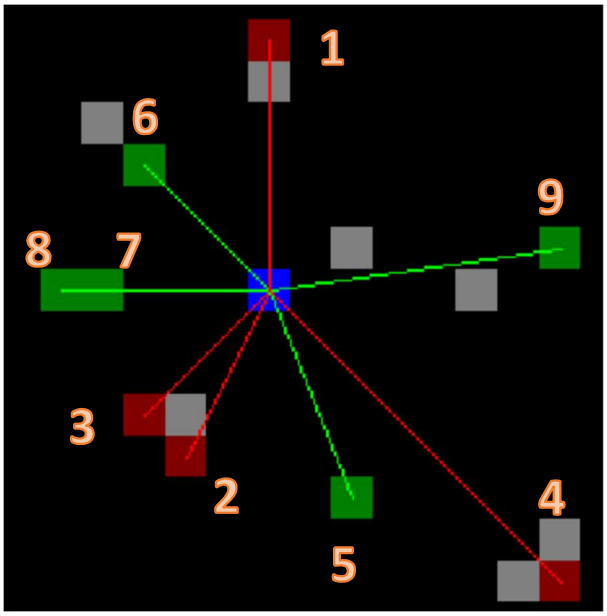
**Output :** Le nombre d'adversaires que vous pourriez toucher avec un headshot (un entier positif). Autrement dit : le nombre de segments reliant le centre de votre case au centre de la case d'un adversaire, tel que ces segments n'intersectent pas avec la hitbox carrée d'un mur.

**Exemple :**
La carte suivante est un carré de 15 cases de côté. Sur la visualisation ci-contre :

- Carré bleu : le joueur « P »
- Carré gris : mur « W »
- Carré vert (et segment vert) : ennemi « E » avec une vision suffisante, à compter
- Carré rouge (et segment rouge) : ennemi « E » suffisamment caché, à ne pas compter
- Numéro d'un adversaire : différents cas possibles. *Parmi ceux-ci,* ***les cas 3 et 4 ne sont pas présents*** *dans le véritable fichier d'entrée, vous n'avez pas besoin de prendre en compte leur particularité.*

Input :

*15*
*FFFFFFFFFFFFFFF*
*FFFFFFFEFFFFFFF*
*FFFFFFFWFFFFFFF*
*FFFWFFFFFFFFFFF*
*FFFFEFFFFFFFFFF*
*FFFFFFFFFFFFFFF*
*FFFFFFFFFWFFFFE*
*FFEEFFFPFFFFWFF*
*FFFFFFFFFFFFFFF*
*FFFFFFFFFFFFFFF*
*FFFFEWFFFFFFFFF*
*FFFFFEFFFFFFFFF*
*FFFFFFFFFEFFFFF*
*FFFFFFFFFFFFFFW*
*FFFFFFFFFFFFFWE*




- 1 est entièrement caché par un mur.
- 2 est partiellement caché par un mur : son centre n'est pas visible.
- *3 est partiellement caché par un mur : son centre n'est tout juste pas visible (le coin du mur est sur la trajectoire).*
- *4 est entièrement caché par deux murs : les coins des deux murs sont sur la trajectoire.*
- 5, 6 et 7 sont visibles.
- **8 est visible** malgré qu'il soit derrière 7. 7 n'est compté qu'une seule fois dans le résultat, malgré qu'il serait aussi touché (même en son centre) par un tir sur 8.
- **9 est visible** parce que **son centre est visible**, malgré les deux murs qui l'encadrent (les murs n'empêchent pas de voir le centre de la case).

Output : 5

# 7 - Fortnite Championship Series

A la fin d'une partie au Fortnite Championship Series, nous souhaitons calculer différentes statistiques afin de pouvoir les présenter au public.

Pour cela, nous avons accès à 4 fichiers générés à la fin de la partie :

- La liste des joueurs et leur équipe correspondante
- La liste des équipes et les différentes zones traversées par l'équipe
- La liste des zones sur la carte et les ressources disponibles dans celle-ci. Lorsque qu'un joueur passe dans cette zone, alors il va récupérer le nombre de ressources indiquées dans la zone
- Le nombre de ressources utilisées par joueur

Ces 4 fichiers sont à votre disposition, que vous pourrez exploiter pour fournir les réponses aux 3 questions suivantes :

1. Quelle est l'équipe ayant dépensé le plus de ressources ?
2. Combien de bois ont été récupérés par toutes les équipes ?
3. Quelle est l'équipe ayant récupéré le moins de ressources ?

**Input :** 4 Fichiers :

1. players.csv : contient le nom des différents joueurs ainsi que leur équipe :
    - player : Nom du joueur.
    - team : Nom de l'équipe dans laquelle le joueur fait partie.

2. teams.csv : contient la liste des équipes ainsi que les zones qu'elles ont traversées :
    - team : Nom de l'équipe.
    - location : Nom de la zone traversée par l'équipe.

3. locations.csv : contient la liste des zones ainsi que les ressources récupérées par zone :
    - location : Nom de la zone.
    - wood : Nombre de bois récupéré dans cette zone.
    - stone : Nombre de pierre récupéré dans cette zone.
    - metal : Nombre de métal récupéré dans cette zone.

4. resources_used.csv : contient les ressources utilisées par chaque joueur :
    - player : Nom du joueur.
    - wood : Nombre de bois utilisé par le joueur.
    - stone : Nombre de pierre utilisé par le joueur.
    - metal : Nombre de métal utilisé par le joueur.

**Output :** Réponse sous la forme «A, B, C» avec :

- A : réponse à la question 1, le nom d'une équipe comme défini dans le fichier teams.csv
- B : réponse à la question 2 (nombre entier)
- C : réponse à la question 3, le nom d'une équipe comme défini dans le fichier teams.csv

**Exemple :**

**Input :**

*players.csv :*

*player1,team1*

*player2,team1*

*player3,team2*

*player4,team2*

*teams.csv :*

*team1,location1*

*team2,location1*

*team2,location2*

*locations.csv :*

*location1,10,15,20*

*location2,15,10,30*

*resources_used.csv :*

*player1,5,8,10*

*player2,3,4,6*

*player3,10,10,25*

*player4,13,5,20*

**Output :** team2, 70, team1

# 8 – Team Rebuild

Lors d'une partie de ***Fortnite***, les joueurs ont la possibilité de discuter au travers d'un chat vocal.

Dans une démarche de traçabilité des échanges et d'analyse, ces échanges sont retranscrits dans un fichier *.csv* que vous avez pu vous procurer.

À partir du fichier fourni en entrée, vous devez retrouver la composition des équipes.

Pour cela, on considère qu'un joueur est dans la même équipe que le joueur qu'il a le plus interpellé.

Une interpellation consiste à avoir le nom complet du joueur dans la transcription (un message).

Une transcription peut contenir plusieurs interpellations, que ce soit pour des coéquipiers ou adversaires.

**Input :** *team_rebuild_brest.csv* au format P,M avec :

- P : nom du joueur ayant parlé
- M : message envoyé (par exemple « Hey JohnDoe66, look out behind you, there's GuyTare35 ! »)

**Output :** Plusieurs listes (correspondant au nombre d'équipes) où :

- Les joueurs d'une équipe sont triés par ordre alphabétique (A à Z)
- Les équipes sont triées par ordre alphabétique du premier joueur (A à Z)

Par exemple : *[BarbeNoire;JyGo;Limonade;...];[Domino;GuyTare;Kimoo;...];....*

# 9 – Sniper Forever

Dans le monde de Fortnite, la précision des tirs est essentielle pour remporter la victoire. Rien de mieux qu'un bon sniper pour éliminer ses adversaires avec élégance. Cependant, les développeurs ont décidé de nous compliquer la tâche en appliquant la gravité aux balles tirées par ces derniers.

Simulez les tirs à partir des données en entrée et déterminez s'ils touchent leur cible. La réponse attendue est la liste des numéros des tirs (en commençant par 1) qui touchent leur cible.

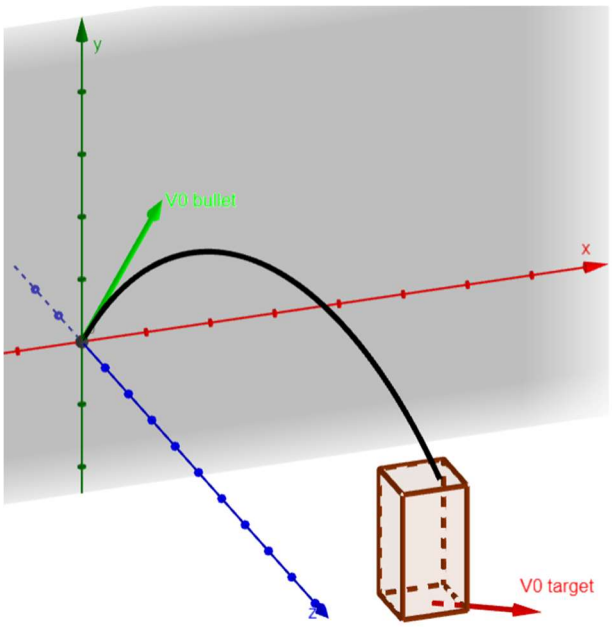
On considèrera que le coup de feu part des coordonnées (0, 0, 0) et la gravité est égale à 9,81 unités/s².

La cible est une hitbox de forme parallélépipède rectangle de 1 unité de côté et 2 unités de hauteur et dont la base est fixée à 0 sur l'axe vertical.

**Input :** Le fichier *sniper_forever.csv* au format v0x_bullet,v0y_bullet,v0z_bullet,v0x_target,v0z_target,x0_target,z0_target avec :

- v0x_bullet : la vitesse initiale du tir projetée sur l'axe x
- v0y_bullet : la vitesse initiale du tir projetée sur l'axe y
- v0z_bullet: la vitesse initiale du tir projetée sur l'axe z
- v0x_target : la vitesse initiale de la cible projetée sur l'axe x
- v0z_target : la vitesse initiale de la cible projetée sur l'axe z
- target_x : la position initiale du centre de la cible sur l'axe x
- target_z : la position initiale du centre de la cible sur l'axe z

Correspondance des données d'entrée :




**Output :** Liste des numéros des tirs (en commençant par 1) qui touchent leur cible, séparés par des points-virgules (ex : 4;7;8;12).

# 10 – Is This A Fair Tournament

Un tournoi de 200 joueurs en solo a récemment pris place avec une formule plutôt particulière. Durant l'évènement, ce sont 100 parties qui ont été jouées. Les joueurs pouvaient jouer autant de parties qu'ils voulaient. Ils gagnent des points pour chaque élimination d'un adversaire ainsi que lors d'un top 1. Les joueurs démarrent avec 0 points.

Les points sont distribués de la manière suivante :

- Un top 1, c'est-à-dire être le dernier survivant d'une partie, octroie 100 points bonus

- Les points gagnés pour chaque élimination dépendent de la différence de points entre le joueur et son adversaire. Il devient alors avantageux d'éliminer les adversaires ayant le plus de points et fuir un combat avec un adversaire ayant moins de points que soi est une option à considérer. Le tableau suivant donne la répartition de ces points :

| Différence min | Différence max | Points gagnés |
|---|---|---|
| -∞ | 5 | 2 |
| 6 | 10 | 4 |
| 11 | 20 | 8 |
| 21 | 50 | 16 |
| 51 | 100 | 32 |
| 101 | 200 | 64 |
| 201 | 500 | 128 |
| 501 | +∞ | 256 |

Le classement final du tournoi ne tient compte que du nombre de points gagnés par les joueurs.

Parmi les joueurs qui ont fait **le plus de top 1**, retrouvez le **nom**, la **place** et le **nombre de points** dans le classement final de celui qui a eu le meilleur score.

**Input :** is_this_a_fair_tournament.txt structuré de la manière suivante :

- L1 : « Players »
- N lignes : liste des joueurs
- Saut de ligne
- 100x le bloc suivant
  - « Game M » (avec M le numéro de la partie)
  - K : nombre de joueurs participants à la partie
  - P lignes au format J1,J2 avec :
    - J1 : joueur qui élimine
    - J2 joueur éliminé
  - Saut de ligne

**Output :** Le nom, la place et le nombre de points du joueur au meilleur score parmi ceux qui ont fait le plus de top 1 sous la forme nom;place;points

# Optimisation – Une bonne marche

C'est une nouvelle journée de travail qui commence pour Gontran l'ambulant. Muni de tout un stock d'armes et autres bricoles, Gontran joue un rôle essentiel dans Fortnite. En effet, c'est lui qui va disposer tout le *loot* aux quatre coins de la carte, avant que la partie ne commence et que les joueurs ne se ruent sur ces précieux objets, luttant pour leur survie.

Gontran aime faire de longues balades et est fier de son travail, et en plus ça paye bien. Précisément, il est payé au nombre de kilomètres parcourus, et cela fait des années qu'il s'arrange pour marcher *longtemps*.

Malheureusement, Gontran a du mal à réfléchir aujourd'hui, et il vous demande donc de prévoir un trajet pour lui. Durant ses préparatifs, il a déjà repéré les points d'intérêts par lesquels il doit passer pour y déposer du loot. Il a noté leurs coordonnées sur un papier qu'il vous tend (le fichier d'entrée).

**Quel est le trajet le plus long, passant exactement une fois par chaque point d'intérêt ? (sans revenir au point de départ)**

Gontran ne s'attend pas à ce que vous lui indiquiez le *meilleur* trajet, mais simplement ***l'un des meilleurs*** trajets possibles. Plus la distance parcourue est grande, et mieux ce sera.

Précisions :

- Vous pouvez partir de n'importe quel point de départ, et terminer le trajet à n'importe quel point d'arrivée. Le point d'arrivée doit être différent du point de départ.
- Tous les points d'intérêts doivent être visités exactement une fois.
- On considère une carte plane, seules les coordonnées X et Y seront prises en compte.
- Le trajet d'un point à un autre se fait en ligne droite, la **distance euclidienne** doit-être utilisée.
    - Exemple pour deux point A(aX ; aY) et B(bX ; bY) , la distance les séparant est :
    - `sqrt( (bX - aX) * (bX - aX) + (bY - aY) * (bY - aY) )`

**Input :** Le fichier ***coordinates.txt*** qui contient les informations ci-dessous.

Une première ligne avec la valeur N (un entier) indiquant le nombre de points d'intérêts à visiter.

Les N lignes suivantes correspondent chacune à un point d'intérêt. Elles suivent le format "n x y" où :

- n (un caractère) : identifiant du point d'intérêt
- x (un entier) : coordonnée X du point d'intérêt
- y (un entier) : coordonnée Y du point d'intérêt

**Output :** Les N identifiants de points d'intérêts, dans l'ordre dans lequel ils sont parcourus (caractères mis bout-à-bout, sans séparateur).

Le meilleur trajet est celui qui maximise la distance parcourue, ce qui correspond à la somme des distances point-à-point dans l'ordre de parcours.

Les distances ne sont pas tronquées (certaines solutions pourront être départagées à la décimale prêt).

**Exemple de réponse :** En considérant seulement 6 points d'intérêts, la réponse "AZERTY" correspond au trajet commençant par A, puis allant à Z, puis E, R, T, et se terminant par Y.

La distance parcourue lors de ce trajet correspond à la somme des distances de A à Z, de Z à E, de E à R, de R à T, et de T à Y.

**Instructions particulières :** Rentrez votre solution (le trajet) sur le site, dans la limite de 5 tentatives par minute. Seule la dernière solution renseignée sera prise en compte.

La page répondra systématiquement que la solution est incorrecte. Une validation sera effectuée manuellement après l'épreuve et des points bonus seront accordés aux équipes ayant produit les meilleures solutions.

Le classement sera effectué selon les critères suivants :

- Taille totale du parcours
- En cas d'égalité, les solutions seront ensuite départagées par l'heure de rendu de la solution