CRC

COMPÉTITION DE PROGRAMMATION

PROBLÈME PRÉLIMINAIRE 2

QUELQUES NOTES

- Les règles complètes sont dans la section 4 du livret des règlements.
- Vous avez jusqu'au **dimanche 1er décembre, 23h59** pour remettre votre code.
- N'hésitez pas à utiliser le forum de programmation sur le discord de la CRC pour poser vos questions et discuter des problèmes. Il est là pour ça!
- **On vous donne des fichiers modèles faciles à utiliser pour votre code et pour faire vos tests. Vous devez les utiliser pour résoudre le problème!**

UTILISATION DU FICHIER MODÈLE

- Le fichier de test appelle la fonction associée avec en paramètre les informations du test et compare sa sortie avec ce qui est attendu pour vous permettre de voir si les tests réussissent. **Tout votre code (sauf fonctions additionnelles que vous créez) devrait être écrit dans la fonction prévue à cet effet.**
- Les points mis dans le document indiquent la difficulté et le pointage attribué pour la réussite pour chaque défi. Ce problème préliminaire aura une valeur globale de 2% du défi principal.

STRUCTURE

Une petite mise en situation comme celle-ci explique les fondements de chaque défi et offre les bases nécessaires pour résoudre celui-ci.

Spécification d'entrée et de sortie:

Contient les caractéristiques des entrées fournies ainsi que les critères attendus pour les sorties du programme.

Exemple d'entrée et de sortie:

Contient un exemple d'entrée, parfois constitué lui-même de plusieurs sous-exemples, pour que vous puissiez tester votre programme. Chaque exemple de sortie donne la réponse attendue pour l'entrée correspondante.

Explication de la première sortie:

Décortique davantage le défi en expliquant comment la première entrée est traitée et en montrant le chemin menant à cette réponse.

Défis félins pour développeurs malins

Bienvenue dans le royaume félin du code, où chaque ligne de programmation ronronne d'ingéniosité ! Votre mission, si vous l'acceptez, est de résoudre une série de défis miaou-gnifiques, chacun ponctué de jeux de mots félins.

Les serveurs sont en **chat-rnade** : il faudra débugger un script qui donne des résultats miaul-cules. Ensuite, plongez dans l'algorithme "GriffeSort", qui classe des données avec une efficacité presque chat-leureuse, mais attention aux "boucles infinies", elles risquent de vous prendre à la patte ! Enfin, préparez-vous à concevoir une **appli-chat-ion** capable de traduire les miaulements en langage humain.

Chaque problème sera un véritable casse-tête, mais souvenez-vous : pour chaque erreur, il y a une solution qui chat-tend juste un peu de réflexion. Alors, sortez vos claviers, affûtez vos compétences et préparez-vous à coder avec élégance féline. Que la créativité et les jeux de mots soient avec vous ! 🐾

Comme vous vous doutez, c'est CHAT GPT qui nous a fait ce beau message. Depuis l'arrivée de CHAT GPT dans nos vies, beaucoup de choses ont changé en programmation, donc voici une belle série de problèmes de CHATs pour vous!

Partie 1: ConCATenation (10 points)

Après les populaires boy math et girl math, nous vous présentons les CAT math. C'est un concept de mathématique alternative dans lesquels, les opérations sont différentes, mais les nombres utilisés sont les mêmes. Nous allons ici seulement explorer les concepts d'addition et de multiplication. Vous devrez faire un programme qui résout des calculs simples de CAT math. En CAT math, le résultat de l'addition de deux nombres en cat math est les deux nombres mis un à la suite de l'autre.

$$x + y = xy$$

Voici un exemple simple:

$$4 + 3 = 43$$

Pour ce qui est de la multiplication, on multiplie deux nombres en CAT math en répétant le second nombre du nombre de fois indiqué par le premier nombre.

$$3 * 4 = 444$$
$$2 * 14 = 1414$$

Spécification d'entrée et de sortie:

En entrée vous recevrez dans une variable *string* une équation qui contient deux nombres séparés par l'opérateur d'addition ou de multiplication. Les nombres fournis seront toujours positifs.
En sortie vous devrez donner le résultat sous forme d'une variable de type *int*.

Exemple d'entrée:

4+11
6*30
178+82
12*3

Exemple de sortie:

```
411
303030303030
17882
333333333333
```

Explication de la première sortie:

Dans la première équation nous avons la valeur 4 additionnée à la valeur 11. Dans ce cas, nous avons seulement à mettre 4 suivis de 11 pour obtenir 411. Pour la première multiplication, on a 6 qui est le nombre de fois qu'on répète le nombre 30 ce qui nous donne 6 fois le nombre 30 donc 303030303030.

Partie 2: CHATpeaux (10 points)

Vous ouvrez une boutique de CHATpeaux haut de forme. Vous devez par contre calculer le profit de vente de vos CHATpeaux. Comme vous les faites sur mesure, vous devez calculer la quantité de matériel nécessaire. Une fois que vous avez les dimensions et le coût d'achat des matières premières, vous devez calculer le profit en vous gardant bien sur une marge de profit!

Le chapeau se compose de trois sections, la première section est le dessus du CHATpeau qui forme un cercle. Vous recevrez donc la largeur de la tête du client qui va être le diamètre du dessus. Voici la formule mathématique pour trouver la quantité de matériel du dessus du chapeau.

$$top = \pi * (\frac{d}{2})^2$$

La deuxième section du chapeau est le cylindre central. Vous devez donc calculer le côté du CHATpeau en fonction de la hauteur que le client veut. Voici la formule pour la quantité de matériel du côté.

$$side = 2 * \pi * (\frac{d}{2}) * h$$

La troisième et dernière partie du CHATpeau est le rebord. Vous devez le faire selon la dimension choisie de votre client. La mesure de la bordure sera représentée par la lettre b. Voici le calcul de la surface du rebord selon la largeur de bordure.

$$brim = \pi * (b + (\frac{d}{2}))^2 - \pi * (\frac{d}{2})^2$$

Vous avez maintenant votre quantité de matériel qui est la somme des différentes sections du chapeau. Une fois que vous avez la quantité de matière voulue, il ne vous reste qu'à multiplier par le coût du matériel et par votre marge de profit. Votre marge de profit va être un pourcentage qui va vous être fourni et qui va varier d'un CHATpeau à l'autre. Vous aurez donc le profit avec l'équation suivante:

$$profit = materials * material\,cost * profit\,margin$$

Spécification d'entrée et de sortie:

En entrée vous recevrez:

- d: un *int* de la largeur de l'ouverture du chapeau
- h: un *int* de la hauteur du chapeau
- b: un *int* de la largeur du rebord du chapeau
- m_cost: un *float* du coût des matériaux
- margin: un *float* du pourcentage de profit que vous voulez faire sur le chapeau

En sortie vous devrez donner une variable *float* avec 2 décimales de précision

Exemple d'entrée:

d = 16, h = 14, b = 4, m_cost = 0.42, margin = 0.1
d = 18, h = 16, b = 3, m_cost = 0.38, margin = 0.15
d = 17, h = 16, b = 5, m_cost = 0.40, margin = 0.18

Exemple de sortie:

48.56
77.36
102.75

Explication de la première sortie:

Avec les valeurs données, on commence par calculer la quantité de matériaux pour la partie du top avec la formule

$$top = \pi * (\frac{d}{2})^2$$

$$top = \pi * (\frac{16}{2})^2$$

$$top = 201.06192982974676$$

On peut maintenant calculer la quantité de matériel pour le side

$$side = 2 * \pi * (\frac{d}{2}) * h$$

$$side = 2 * \pi * (\frac{16}{2}) * 14$$

$$side = 703.7167544041137$$

Maintenant, nous pouvons calculer la quantité de matériel pour le brim

$$brim = \pi * (b + (\frac{d}{2}))^2 - \pi * (\frac{d}{2})^2$$

$$brim = \pi * (4 + (\frac{16}{2}))^2 - \pi * (\frac{16}{2})^2$$

$$brim = 251.32741228718345$$

Une fois que nous avons calculé la quantité de matériel nécessaire, nous pouvons calculer le profit avec le coût du matériel et la quantité de matériel.

$$profit = materials * material\,cost * profit\,margin$$

$$profit = 1156.1060965210438 * 0.42 * 0.1$$

$$profit = 48.55645605388384$$

$$profit = 48.56$$

Partie 3: CaptCHAT (60 points)

Le CAPTCHA est une mesure de sécurité conçue pour empêcher les cyberattaques en vérifiant qu'un internaute est bien un humain et non un bot. Un test CAPTCHA peut consister en une série aléatoire de lettres et/ou de nombres à identifier ou même un problème de maths à résoudre. Malheureusement, plusieurs chercheurs ont développé des algorithmes capables de réussir les CAPTCHA classiques. Vous décidez donc de créer CaptCHAT, un test CAPTCHA amélioré!

Le CaptCHAT comporte une image ascii d'un chat qui représente deux séquences de texte. Vous allez écrire un programme qui transforme deux séquences de texte en image ascii. C'est à partir de ces images qu'un internaute doit déchiffrer le texte original!

Voici les CAT (quatre) images par défaut du CaptCHAT, chacune ayant une **hauteur de 3 lignes** et une **largeur de 5 colonnes**, séparées par des dièses:

```
 ^-^  #  ^-^  #  ^-^  #  ^-^
|'_'| # |-_-| # |*_*| # |@_@|
 > <  #  > <  #  > <  #  > <
```

Vous allez modifier ces images en fonction de la **différence entre 2 séquences de texte** grâce à la **distance de Levenshtein**.

La **distance de Levenshtein** mesure le nombre minimal d'opérations nécessaires pour transformer une chaîne de caractères (*chaîne1*) en une autre (*chaîne2*). Les **3 opérations autorisées** sont l'ajout, la suppression et le remplacement.

- Ajout (ou insertion): insérer un caractère dans *chaîne1*.
    - chat → chats: on insère la lettre 's' dans la chaîne "chat".
- Suppression (ou effacement): supprimer un caractère dans *chaîne1*.
    - chats → cats: on supprime la lettre 'h' dans la chaîne "chats".
- Remplacement (ou substitution): remplacer un caractère par un autre dans *chaîne1*.
    - cats → hats: on remplace la lettre 'c' par 'h' dans la chaîne "cats".

Notons pourtant que la distance de Levenshtein entre "chat" et "hats" **n'est pas** 3. En effet, on a seulement besoin de 2 opérations pour transformer "chat" en "hats": on peut, par exemple, supprimer d'abord la lettre 'c' ("chat" → "hat"), puis ajouter la lettre 's' à la fin ("hat" → "hats"). Le nombre **minimal** d'opérations nécessaires, c'est-à-dire la distance de Levenshtein entre "chat" et "hats", est donc 2.

Voici la définition formelle de la distance de Levenshtein $lev(a,\ b)$ entre deux chaînes *a* (ayant $|a|$ lettres) et *b* (ayant $|b|$ lettres):

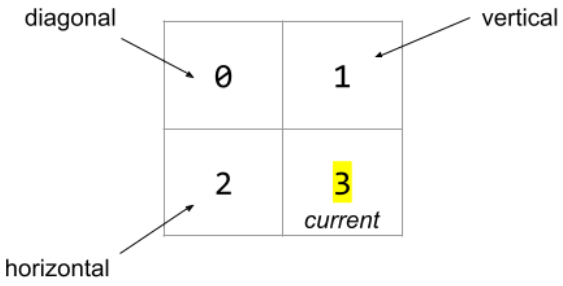
$$\operatorname{lev}(a,b) = \begin{cases} \max(|a|,|b|) & \text{si } \min(|a|,|b|) = 0, \\ \operatorname{lev}(a-1,b-1) & \text{si } a[0] = b[0], \\ 1 + \min \begin{cases} \operatorname{lev}(a-1,b) \\ \operatorname{lev}(a,b-1) \\ \operatorname{lev}(a-1,b-1) \end{cases} & \text{sinon.} \end{cases}$$

On peut construire une matrice avec les distances entre toutes les sous-chaînes de *chaîne1* et *chaîne2*. La valeur dans **le coin en bas à droite** est la distance de Levenshtein entre *chaîne1* et *chaîne2*. Voici la matrice pour “chat” et “hats”:

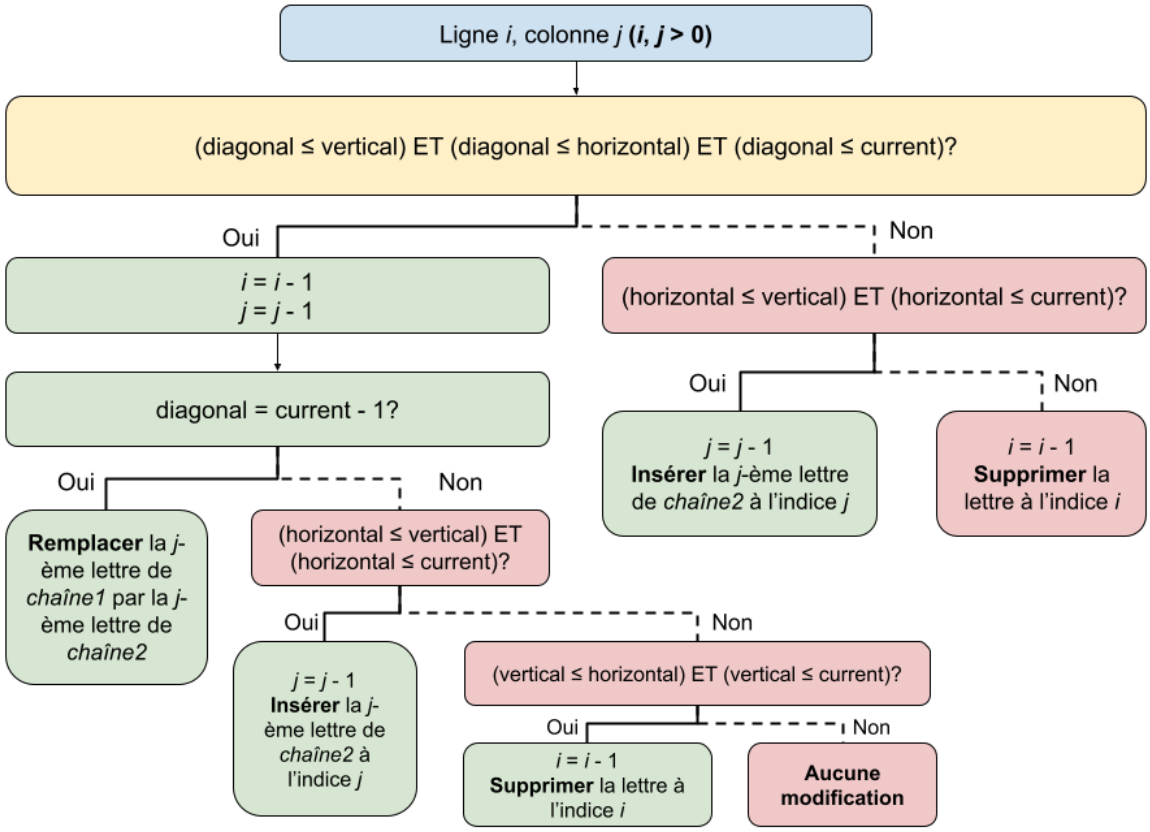
		h	a	t	s
	0	1	2	3	4
c	1	1	2	3	4
h	2	1	2	3	4
a	3	2	1	2	3
t	4	3	2	1	2

Il y a souvent plusieurs façons optimales de transformer *chaîne1* en *chaîne2*. Afin d’assurer qu’on obtient toujours les mêmes opérations, on va utiliser un algorithme de **retour sur trace** (ou retour arrière, appelé aussi *backtracking* en anglais).

On commence dans le coin en bas à droite de la matrice de distances ayant *m* lignes (0, 1, 2, …, $m-1$) et *n* colonnes (0, 1, 2, …, $n-1$). Définissons les éléments *diagonal*, *vertical*, et *horizontal* par rapport à la valeur actuelle (*current*):



Voici comment déterminer l'opération effectuée pour la valeur de la *i*-ème ligne et de la *j*-ème colonne (la valeur actuelle, *current*) de la matrice de distances:



L'algorithme se termine lorsqu'on atteint la ligne 0 ou la colonne 0.

Après avoir trouvé les opérations utilisées pour transformer *chaîne1* en *chaîne2*, vous devez modifier l'image CaptCHAT de la même façon. Vous allez effectuer les ajouts sur la ligne 1, les suppressions sur la ligne 2, et les remplacements sur la ligne 3. Vu que les CaptCHAT par défaut ont une largeur de 5 colonnes, si l'opération a été effectuée sur un indice $ind \geq 5$, utilisez à la place le reste de la division $ind \div 5$.

Enfin, la distance de Levenshtein $dist$ entre chaîne1 et chaîne2 déterminera la forme des yeux du CaptCHAT. Selon le reste de la division $dist \div 4$, les yeux seront:

- `' '` si le reste = 0,
- `- -` si le reste = 1,
- `* *` si le reste = 2,
- `@ @` si le reste = 3.

Spécification d'entrée et de sortie:

En entrée, vous recevrez un *array* avec deux variables de type *string* que vous devez convertir en image ascii. Les chaînes fournies seront toujours en minuscules.

En sortie, vous donnerez un *array* formant l'art ascii du chat correspondant au texte d'entrée.

Exemples d'entrée:

```
["kitten", "sitting"]

["chatons", "adorables!"]

["pun'ctu/ation", "pon~ctuat'ion"]
```

Exemples de sortie:

```
[" g^-^ ",
 "|@_@|",
 "s> <i"]

[" ^-r!^ ",
 "|'_'| ",
 "ble<  "]

[" ^-^' ",
 "|''| ",
 " o ~ "]
```

Explication de la première sortie:

On veut transformer "kitten" en "sitting". Voici la matrice de distances:

		s	i	t	t	i	n	g
	0	1	2	3	4	5	6	7
k	1	1	2	3	4	5	6	7
i	2	2	1	2	3	4	5	6
t	3	3	2	1	2	3	4	5
t	4	4	3	2	1	2	3	4
e	5	5	4	3	2	2	3	4
n	6	6	5	4	3	3	2	3

Utilisons maintenant le retour sur trace. On commence par la valeur dans le coin en bas à droite, la 6e ligne et la 7e colonne (*i* = 6, *j* = 7):

```
      s  i  t  t  i  n  g
   0  1  2  3  4  5  6  7
k  1  1  2  3  4  5  6  7
i  2  2  1  2  3  4  5  6
t  3  3  2  1  2  3  4  5
t  4  4  3  2  1  2  3  4
e  5  5  4  3  2  2  3  4
                     <--   Ajouter 'g' à l'indice 6
n  6  6  5  4  3  3  2  3
```

Il faut comparer les éléments *diagonal*, *vertical*, et *horizontal* selon l'algorithme de retour sur trace. Ici, on a *diagonal* = 3, *vertical* = 4, et *horizontal* = 2. On voit que (*diagonal* ≤ *vertical*), mais (*diagonal* > *horizontal*). Cependant, (*horizontal* ≤ *vertical*) et (*horizontal* ≤ *current*), alors on soustrait 1 de *j* (*j* a maintenant une valeur de 6): l'opération effectuée est donc l'insertion de 'g' à l'indice 6.

L'algorithme continue jusqu'à ce qu'on atteigne l'indice 0 de chaîne1 ou de chaîne2 (c'est-à-dire la ligne 0 ou la colonne 0). Voici toutes les opérations trouvées par le retour sur trace:

```
   0  1  2  3  4  5  6  7
    <--                    Remplacer par 's' à l'indice 0
k  1  1  2  3  4  5  6  7
       \                   Aucune modification à l'indice 1
i  2  2  1  2  3  4  5  6
          \                Aucune modification à l'indice 2
t  3  3  2  1  2  3  4  5
             \             Aucune modification à l'indice 3
t  4  4  3  2  1  2  3  4
                <--        Remplacer par 'i' à l'indice 4
e  5  5  4  3  2  2  3  4  Aucune modification à l'indice 5
                   \  <--  Ajouter 'g' à l'indice 6
n  6  6  5  4  3  3  2  3
```

Les opérations effectuées sont: remplacement par 's' à l'indice 0, remplacement par 'i' à l'indice 4, et ajout de 'g' à l'indice 6. Construisons maintenant l'image CaptCHAT!

La distance de Levenshtein est 3 et le reste de la division $dist \div 4 = 3 \div 4$ est 3, donc les yeux du CaptCHAT sont `@ @`. Notre image CaptCHAT sans modifications est la suivante:

```
[" ^-^ ",
 "|@_@|",
 " > < "]
```

On effectue les ajouts sur la ligne 1. On doit ajouter la lettre ‘g’ à l’indice 6, mais 6 ≥ 5, donc on l’ajoute à l’indice 1 puisque le reste de la division $6 \div 5$ nous donne 1. La ligne 1 devient: `" g^-^ "`

Il n’y a pas de suppressions, donc on ne modifie pas la ligne 2. Sur la ligne 3, on remplace l’espace à l’indice 0 par ‘s’ et l’espace à l’indice 4 par ‘i’. Cela donne l’image finale:

```
[" g^-^ ",
 "|@_@|",
 "s> <i"]
```

Partie CAT: On CHATvire! (30 points)

Vous décidez de faire un tour de bateau en groupe. Par contre, vous détestez l'eau et ne voulez absolument pas CHATvirer. Vous devez donc vous assurer que le bateau reste balancé et pour vous en assurer vous devrez calculer ou vous mettre pour que le centre de gravité de tous s'équilibrent.

Le bateau fait 1 mètre de largeur au total, donc pour noter la position par rapport au centre les positions seront entre -0.5 et 0.5 mètre. Vous recevrez la position et le poids de chacun des passagers. En prenant en compte tous les poids, vous devrez trouver ou vous mettre pour équilibrer le bateau. **Votre poids est toujours de 100 dans le problème.**

Vous devrez donc donner l'endroit où vous placer entre -0.5 et 0.5 avec une précision de deux décimales après la virgule. Par contre, si cet endroit est en dehors du bateau, vous devez retourner la phrase: "On CHATvire!". Pour vous aider à balancer le bateau voici la formule qui doit être respectée pour vous assurer de ne pas CHATvirer.

$$côté\ gauche\ =\ côté\ droit$$

$$\sum_{p\ \text{à}\ gauche} effet\ passager\ \text{à}\ gauche\ =\ \sum_{p\ \text{à}\ droite} effet\ passager\ \text{à}\ droite$$

$$\sum_{p\ \text{à}\ gauche} poids(p_g)\ *\ distance\ du\ centre(p_g)\ =\ \sum_{p\ \text{à}\ droite} poids(p_d)\ *\ distance\ du\ centre(p_d)$$

Ainsi l'effet de vos passagers à gauche doit être égal à celui des passagers à droite et c'est en choisissant ou vous vous placerez dans le bateau pour l'empêcher de CHATvirer. SI l'effet est trop grand pour que vous puissiez le compenser en étant dans le bateau, vous devrez retourner "On CHATvire!"

Spécification d'entrée et de sortie:

En entrée, vous recevrez:

- weights: un array des poids des passagers
- positions: un array des positions par rapport au centre du bateau

En sortie, vous devez fournir un nombre flottant (float) avec une précision de 2 chiffres après la virgule ou si la position n'est pas dans le bateau vous devez retourner la phrase: "On CHATvire!"

Exemple d'entrée:

weights = [0.5, -0.2, 0.5, -0.1]
positions = [40, 70, 70, 40]

weights = [0.2, -0.2, 0.2, -0.2]
positions = [40, 70, 70, 40]

weights = [0.5, -0.2, 0.5, -0.3, 0.4]
positions = [40, 70, 70, 40, 110]

Exemple de sortie:

-0.37
0.0
"On CHATvire!"

Explication de la première sortie:

On veut balancer le bateau donc on calcule l'effet de chacun des côtés.

$$effet\ gauche\ = \sum_{p\ \text{à}\ gauche} effet\ passager\ \text{à}\ gauche$$

$$effet\ gauche\ = \sum_{p\ \text{à}\ gauche} poids(p_g)\ *\ distance\ du\ centre(p_g)$$

$$effet\ gauche\ = 0.2\ *\ 70\ +\ 0.1\ *\ 40$$

$$effet\ gauche\ = 18$$

$$effet\ droite\ = \sum_{p\ \text{à}\ droite} effet\ passager\ \text{à}\ droite$$

$$effet\ droite\ = \sum_{p\ \text{à}\ droite} poids(p_d)\ *\ distance\ du\ centre(p_d)$$

$$effet\ droite\ = 0.5\ *\ 40\ +\ 0.5\ *\ 70$$

$$effet\ droite\ = 55$$

Pour balancer l'effet à gauche et l'effet à droite, on doit choisir une distance à gauche qui produit un effet de 18 - 55 donc un effet de -37. Comme notre poids est de 100 on obtient le calcul suivant.

$$100\ *\ distance\ =\ -\ 37$$

$$distance\ =\ -\frac{37}{100}$$

$$distance\ =\ -\ 0.37$$

Partie 5: CATapultes (30 points)

Lors d'une opération d'assaut d'un CHATeau médiéval, vous êtes chargé de vous occuper des défenses à l'aide d'une CATapulte. Nous allons vous donner la hauteur du rempart où la catapulte est placée, ainsi que la distance horizontale de votre cible. Pour un angle(θ) donné du projectile, vous devrez trouver la vitesse à laquelle l'envoyer. **L'angle vous sera donné en degrés, faites attention avec les conversions en radians!**

Pour ce faire, vous devez utiliser la dynamique de la physique. **Pour vous simplifier la vie, la résistance de l'air est considérée comme nulle dans vos calculs**.

Pour trouver la vitesse initiale, vous vous rappelez de vos cours de physique dans lesquels vous avez vu que le problème peut être décomposé selon deux axes indépendants.

Le premier axe est de gauche à droite donc la distance entre le rempart et la cible. Le second axe est de haut en bas et c'est celui qui est influencé par la gravité. **Dans ce problème, la force de gravité à utiliser est 9.81**.

La vitesse initiale peut être retrouvée en utilisant deux formule, une première pour l'axe des y (la hauteur) et une seconde pour l'axe des x (la distance):

$$0 = h + v_i * sin(\theta) * t - g * t^2$$

$$d = v_i * cos(\theta) * t$$

Avec ces formules, en faisant quelques manipulations et en isolant v_i vous pouvez donc avoir la vitesse initiale avec la formule suivante:

$$v_i = \sqrt{\frac{gd^2}{2cos^2(\theta)*(h+d*tan(\theta))}}$$

Note: $cos^2(\theta) = (cos(\theta))^2$

C'est en implémentant cette formule que vous obtiendrez la vitesse initiale du lancé du projectile. Vous devrez retourner la vitesse initiale avec une précision de deux décimales après la virgule.

Spécification d'entrée et de sortie:

En entrée vous recevrez:

- h: la hauteur du rempart
- d: distance de la cible
- angle: l'angle de lancé du projectile par rapport à l'horizontale

En sortie vous devrez donner la vitesse initiale du projectile sous forme de *float* avec une précision de 2 décimales.

Exemple d’entrée:

h = 160, d = 140, angle = 35
h = 18, d = 100, angle = 40
h = 28, d = 85, angle = 45

Exemple de sortie:

23.56
28.64
25.04

Explication de la première sortie:

Pour trouver la vitesse, on remplace les variables reçues dans l'équation pour obtenir notre vitesse initiale du lancé:

$$v_i = \sqrt{\frac{gd^2}{2cos^2(\theta)*(h+d*tan(\theta))}}$$

$$v_i = \sqrt{\frac{9.81*140^2}{2cos^2(35^o)*(160+140*tan(35^o))}}$$

$$v_i = 23.56398382$$

$$v_i = 23.56$$