

[CYBER1][2024-2025] Partiel (Sujet A)
Algorithmique 2**NOM :** _____**PRÉNOM :** _____

Vous devez respecter les consignes suivantes, sous peine de 0 :

- | | |
|--|--|
| I) Lisez le sujet en entier avec attention | V) Écrivez lisiblement vos réponses (si nécessaire en majuscules) |
| II) Répondez sur le sujet | |
| III) Ne trichez pas | VI) Vous devez écrire les algorithmes et structures en langage C (donc pas de Python ou autre) |
| IV) Ne détachez pas les agrafes du sujet | |

1 Arbres Binaires (8 points)

Questions (6 points)

1.1 Dessinez un arbre répondant aux critères imposés (2 points)

Les arbres ne doivent pas être des arbres vides

1.1.1 (0,5 point) Arbre binaire complet

(hauteur minimale : 3)

1.1.2 (0,5 point) Arbre filiforme

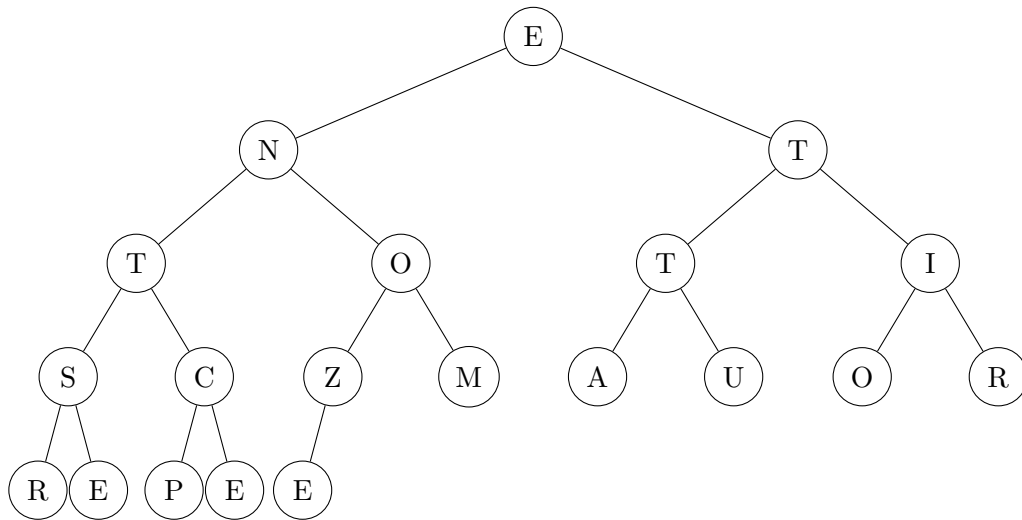
(hauteur minimale : 3)

1.1.3 (0,5 point) Arbre binaire presque complet et localement complet

(hauteur minimale : 0)

1.1.4 (0,5 point) Arbre binaire parfait

(hauteur minimale : 3)

1.2 Répondez aux différentes questions concernant l'arbre suivant (4 points)**1.2.1 (1,5 point) Indiquez toutes les propriétés que possède cet arbre :**

Arité :

Taille :

Hauteur :

Nb feuilles :

☐

Arbre binaire strict / localement complet

☐

Arbre binaire (presque) complet

☐

Arbre binaire parfait

☐

Arbre filiforme

☐

Peigne gauche

☐

Peigne droit

1.2.2 (2 points) Écrivez les clés lors d'un parcours profondeur main gauche de l'arbre dans les 3 ordres ainsi que lors d'un parcours largeur :

Parcours profondeur :

ordre préfixe : _ _ _ _ _

ordre infixé : _ _ _ _ _

ordre suffixe : _ _ _ _ _

Parcours largeur :

ordre : _ _ _ _ _

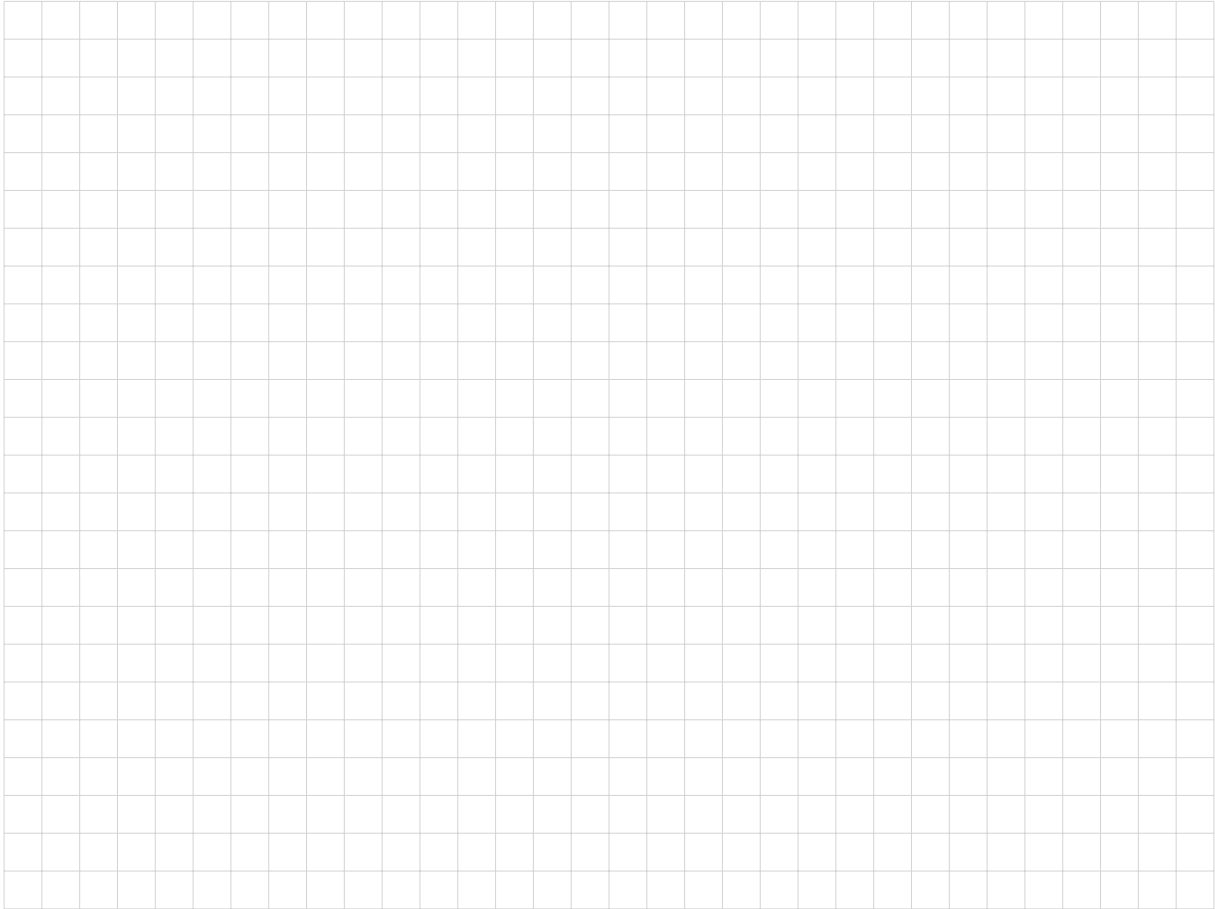
1.2.3 (0,5 point) Indiquez la profondeur et le numéro hiérarchique des nœuds suivants :

	Profondeur	N° hiérarchique
Z		

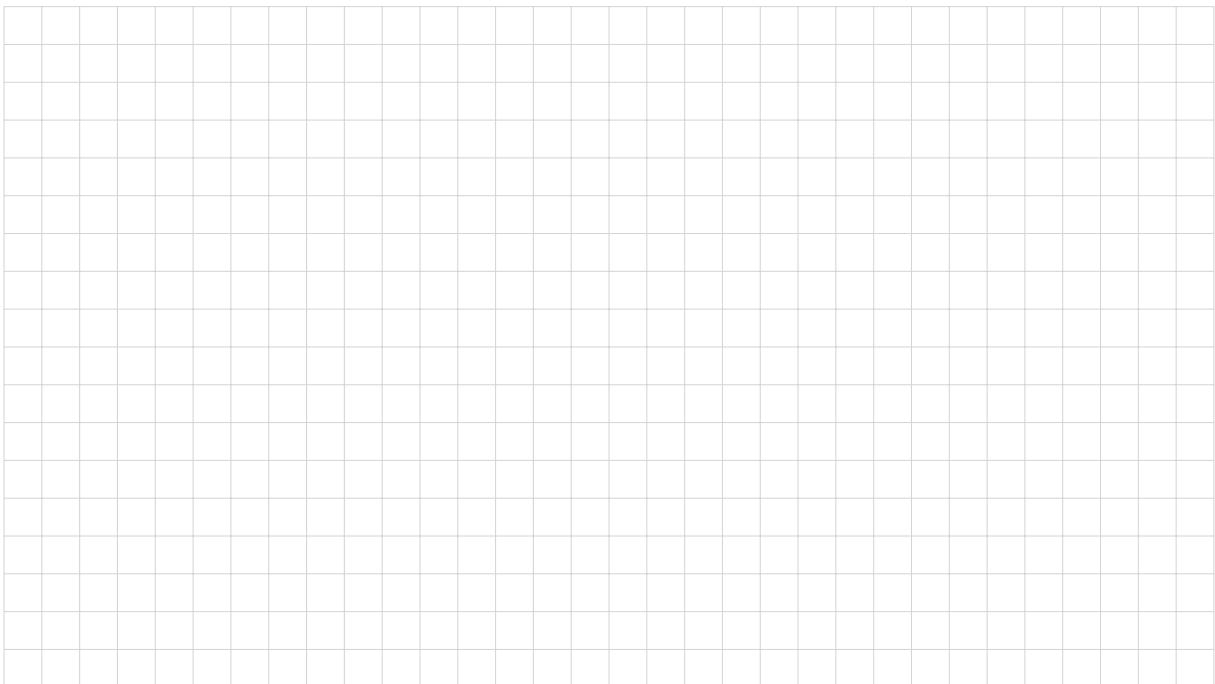
	Profondeur	N° hiérarchique
P		

Algorithmes (2 points)

- 1.3 (1 point) Écrivez une procédure « *BFS* » effectuant un parcours largeur dans un arbre binaire, et affichant les nœuds (l'arbre est de type *node**) :



- 1.4 (1 point) Écrivez une procédure « *DFS* » effectuant un parcours profondeur main gauche dans un arbre binaire, et affichant les nœuds dans les 3 ordres (l'arbre est de type *node**) :



2 Arbres Binaires de Recherche (8 points)

Questions (4 points)

2.1 (4 points) Dessinez le résultat des opérations successives :

- *InsertLeaf*(*node* **R*, *int* *val*) insère l'élément « *val* » en feuille dans l'ABR « *R* »
- *InsertRoot*(*node* **R*, *int* *val*) insère l'élément « *val* » en racine dans l'ABR « *R* »
- *RemoveFromBST*(*node* **R*, *int* *val*) supprime l'élément « *val* » de l'ABR « *R* »

Étape 1
 $R = \text{InsertLeaf}(\text{NULL}, 42)$

Étape 2
 $R = \text{InsertLeaf}(R, 24)$

Étape 3
 $R = \text{InsertLeaf}(R, 64)$

Étape 4
 $R = \text{InsertLeaf}(R, 32)$

Étape 5
 $R = \text{InsertLeaf}(R, 56)$

Étape 6
 $R = \text{InsertLeaf}(R, 36)$

Étape 7
 $R = \text{InsertRoot}(R, 34)$

Étape 8
 $R = \text{InsertLeaf}(R, 72)$


Étape 9 $R = \text{InsertRoot}(R, 38)$ *Étape 10* $R = \text{RemoveFromBST}(R, 42)$

Étape 11 $R = \text{RemoveFromBST}(R, 38)$ *Étape 12* $R = \text{InsertRoot}(R, 60)$

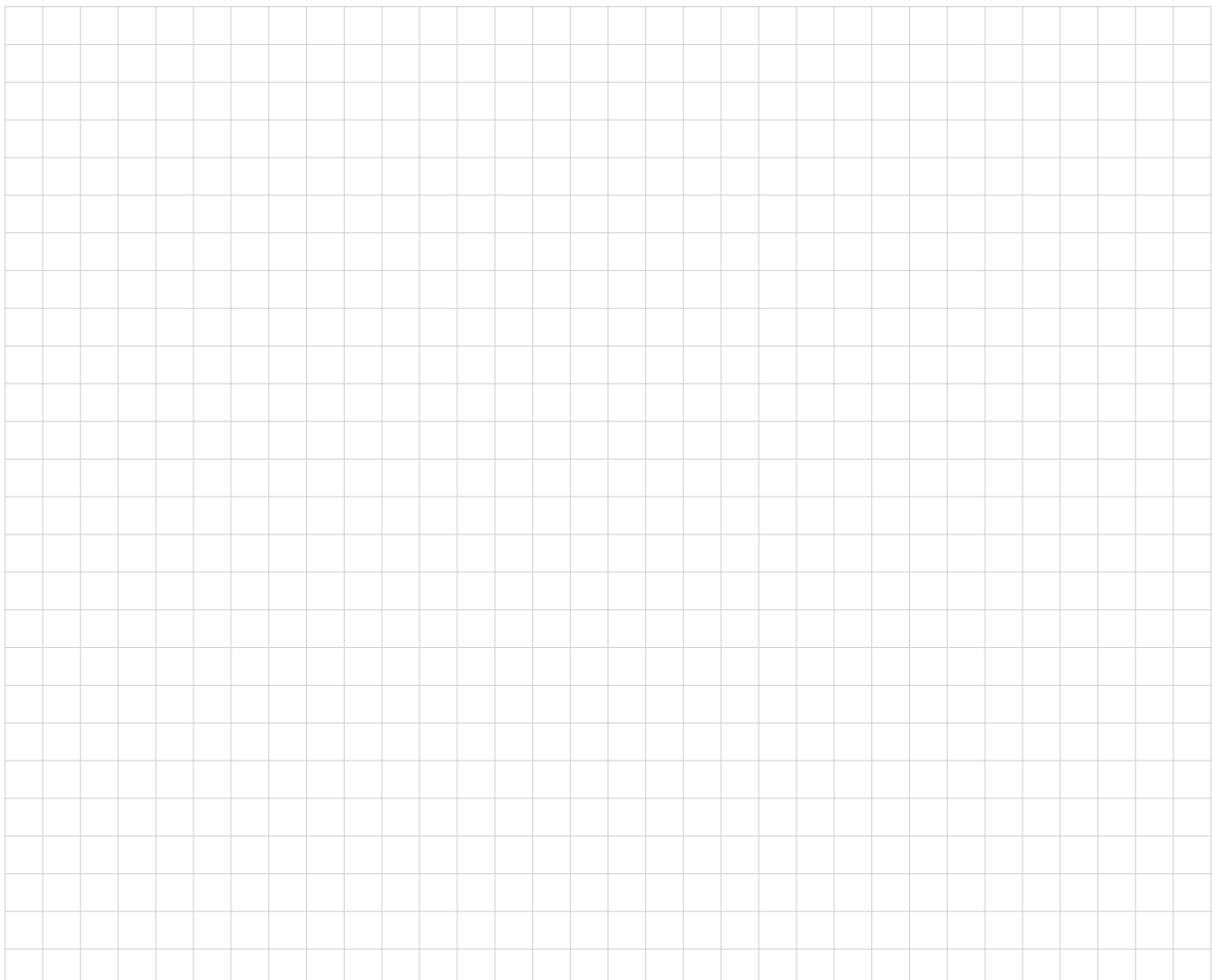
(Espace libre pour dessiner) - Attention : il reste encore des exercices après !

Algorithmes (4 points)

- 2.2 (2 points)** Écrivez une fonction *BSTSearch* recherchant un entier dans un arbre binaire de recherche et renvoyant l'adresse de son nœud. Si l'élément n'est pas trouvé, la fonction doit renvoyer **NULL** :



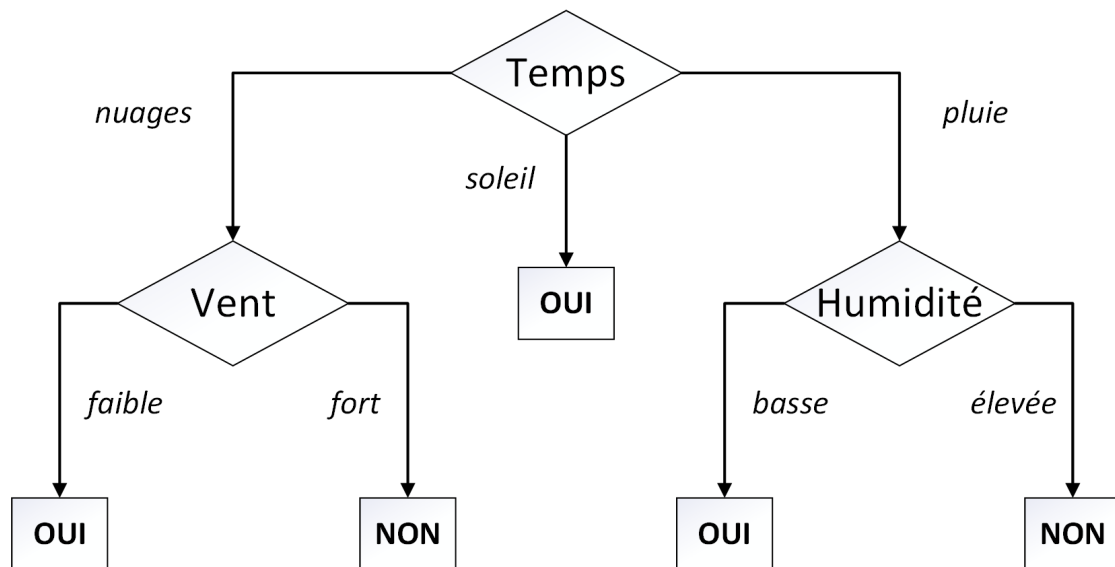
- 2.3 (2 points)** Écrivez une fonction *InsertLeaf* ajoutant en feuille un élément dans un arbre binaire de recherche :



Problème

3 Arbres de Décision (4 points)

Dans le monde de l'intelligence artificielle, il existe de très (trop) nombreux sous-domaines, dont le *décisionnel* ou *aide à la décision* impliquant d'analyser des données pour orienter les décisions, voire, permettre une prise de décision automatique. Parmi les nombreuses méthodes proposées dans la science des données, l'une se prénomme *arbre de décisions*. Comme leur nom le laisse entendre : il s'agit d'arbres, mais leur spécificité réside dans le fait qu'ils représentent les données analysées et permettent d'en déduire une décision.



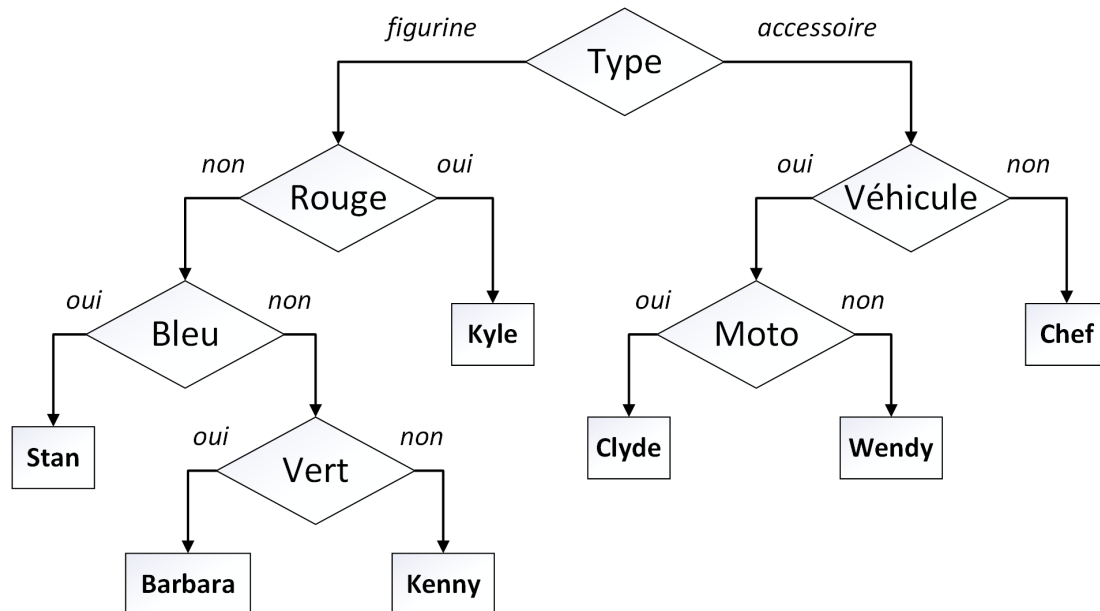
L'exemple le plus célèbre concerne la décision d'aller jouer (ou non) au tennis selon la météo. Dans l'exemple précédent, si le temps est *nuages* et la vitesse du vent est *faible*, ou plus simplement s'il fait *soleil*, alors on sortira jouer au tennis. À l'inverse, si le temps est *pluie*, avec une humidité *élevée*, alors on préférera une autre activité.

Les arbres de décisions sont des arbres dits *généraux*, car leur arité n'est pas nécessairement binaire. Dans le problème que vous traiterez, les arbres seront néanmoins binaires pour vous permettre d'écrire les algorithmes.

[S01E08] Damien

L'anniversaire d'Éric Cartman approche à grands pas, mais, le pauvre Kenny McCormick vient d'être transformé en ornithorynque par Damien (le nouveau de la classe) ! Éric souhaitant à tout prix obtenir sa collection de jouets MegaMan, il décide de consulter la liste des cadeaux qu'il a exigés à ses « amis » afin de changer leur ordre (un ornithorynque ne pouvant *contracter*, il ne peut donc ni effectuer d'achat au magasin de jouets ni payer de TVA à l'État).

Votre travail consiste donc à devoir (malheureusement) aider le jeune Éric Cartman à changer l'assignation des cadeaux de ses « amis » en analysant l'arbre de décision suivant :

**3.1 (2 points) Lecture d'un arbre de décision**

Indiquez qui doit apporter quel cadeau à partir de l'arbre de décision précédent.

Jouet	Caractéristiques	Ami(e) qui doit l'apporter
MegaMan Rouge	<i>figurine, rouge</i>	
MegaMan Vert	<i>figurine, vert</i>	
MegaMan Bleu	<i>figurine, bleu</i>	
MegaMan Jaune	<i>figurine, jaune</i>	
Mega Bécane	<i>accessoire, véhicule, moto</i>	
Mega Turbo Copter	<i>accessoire, véhicule, hélicoptère</i>	
Petite Maison sur la Plage	<i>accessoire</i>	

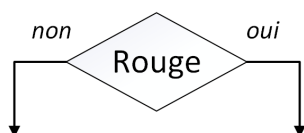
3.2 (2 points) Algorithmme

À partir de la structure suivante (représentant l'arbre de décision), et en réutilisant les fonctions suivantes, écrivez une fonction **get_friend** lisant progressivement une chaîne de caractères contenant les caractéristiques associées à un arbre de décisions afin de renvoyer le nom de l'ami(e) associé(e) à ces caractéristiques. Les caractéristiques seront toujours dans le bon ordre et existeront (il n'y aura pas de cas tordu).

Exemples d'usages et de sorties

Chaîne en entrée	Sortie attendue
"accessoire non"	Chef
"accessoire oui oui"	Clyde
"figurine non oui"	Stan
"figurine non non non"	Kenny

Structure



Wendy

```
struct d_node {
    char *carac; // Caracteristique : Rouge
    char *etq_fg; // Etiquette Fils Gauche : non
    struct d_node *fg;
    char *etq_fd; // Etiquette Fils Droit : oui
    struct d_node *fd;
};
```

```
struct d_node {
    char *carac; // Caracteristique : Wendy
    char *etq_fg;
    struct d_node *fg; // fg = NULL
    char *etq_fd;
    struct d_node *fd; // fd = NULL
};
```

Fonctions autorisées

```
int strlen(char *str);
int strcmp(char *str1, char *str2);
char *get_next_token(char *str, int index);
```

- **strlen(str)** : retourne la longueur de la chaîne *str*
- **strcmp(str1, str2)** : retourne 0 si la chaîne *str1* est égale à la chaîne *str2*
- **get_next_token(str, index)** : extrait une sous-chaîne dans *str* à partir de *index* jusqu'à l'espace suivant ou jusqu'à la fin de chaîne (la sous-chaîne est allouée dans un espace mémoire à part avec malloc). Lorsque la fonction atteint un '**\0**' ou que *str* est **NULL**, elle renvoie **NULL**.

```
char *get_friend(d_node *root, char *str)
```

3.3 (0 point) That's all Folks !

Félicitations, vous avez atteint la fin du partiel et terminé l'année ! Vérifiez bien d'avoir écrit vos nom et prénom sur la première page, et s'il vous reste du temps... n'hésitez pas à dessiner un Cartman ou un Kenny ornithorynque ou autre chose (tant que cela reste correct et n'entraînera donc aucun EpiSignalement).

SUJET A
ALGORITHMIQUE 2