RÉPUBLIQUE
FRANÇAISE

Liberté
Égalité
Fraternité

Concours général du collège

Exemple de sujet pour l'épreuve

d'informatique, technologie et numérique

Durée 2 heures 30 minutes

Composition du sujet

Le sujet est composé de deux parties indépendantes d'égale importance :

- une partie « étude de l'objet technique »
- une partie « algorithmique »

Les deux parties sont indépendantes et peuvent être traitées dans l'ordre souhaité.

Convention d'écriture de programme

Les programmes attendus dans ce sujet doivent être rédigés dans un langage simplifié en français, appelé *pseudocode* dans ce sujet. Ces programmes sont composés de fonctions qui ont un nom, prennent en entrée des valeurs et renvoient un résultat. Une fois une fonction écrite, on peut l'utiliser en y faisant appel dans la suite. On présente ici deux exemples qu'il faut suivre dans votre rédaction.

Une première fonction dont le nom est `suivant` qui prend en entrée un nombre `x`. Cette fonction renvoie `x/2` quand `x` est pair et `3x+1` quand `x` est impair.

```
fonction suivant(x)
  si x est pair
    renvoyer x / 2
  sinon
    renvoyer 3 * x + 1
```

Une seconde fonction dont le nom est `somme` qui prend en entrée un nombre `n`. Cette fonction renvoie la somme des nombres compris entre 1 et `n`. Pour affecter une valeur à une variable, on pourra écrire au choix `Mettre la variable x à 12` ou `x <- 12`.

```
fonction somme(n)
  s <- 0
  pour i allant de 1 à n faire
    s <- s + i
  renvoyer s
```

Ainsi, l'appel `somme(3)` renverra la valeur `6` (car $1 + 2 + 3 = 6$) et la variable `s` contiendra la valeur 6 après l'instruction :

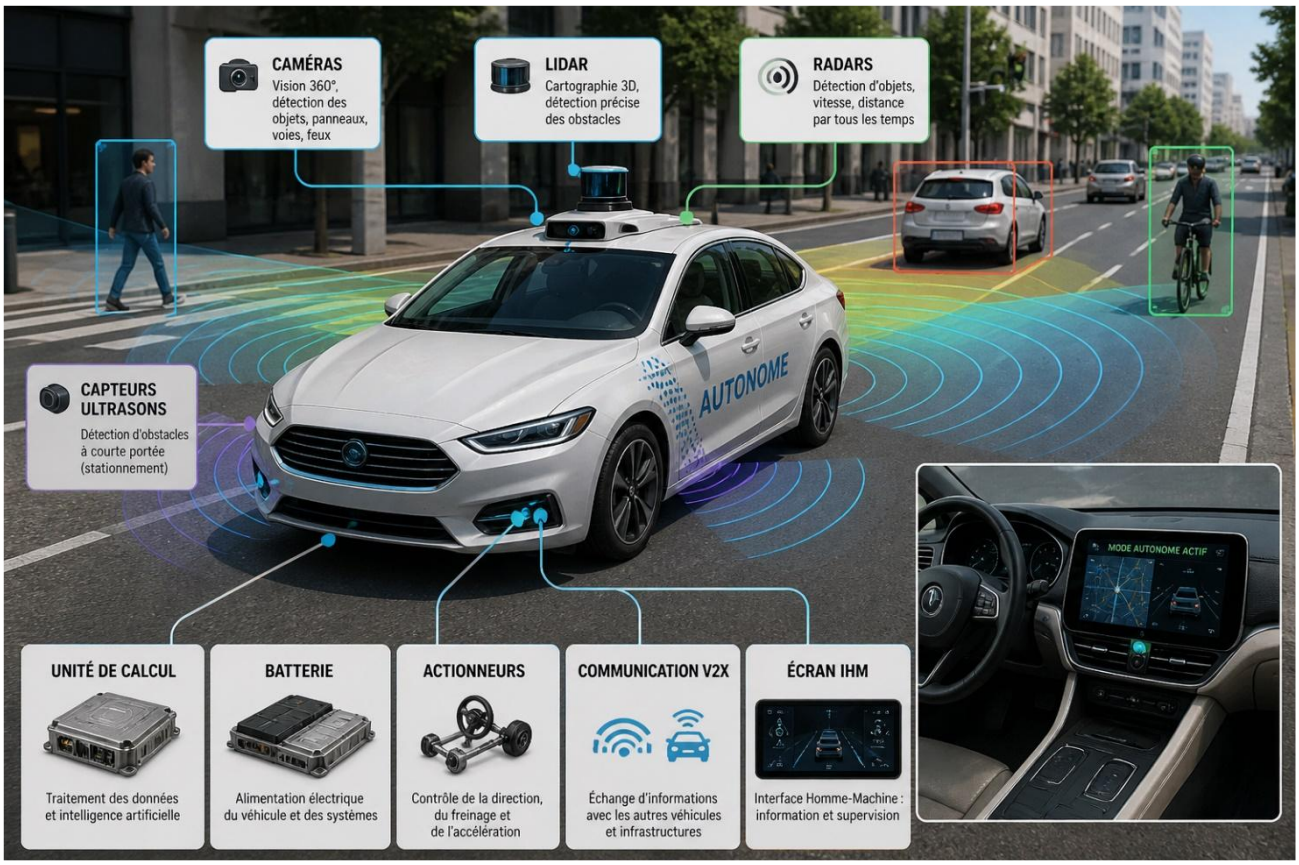
```
s <- somme(3)
```

La syntaxe précise des programmes n'est pas importante ; ce qui compte est d'être clair et compréhensible.

Présentation du sujet d'étude

Les véhicules automobiles autonomes représentent une révolution technologique qui tend à se développer. Le déploiement massif de ce type de véhicules promet une transformation de la mobilité tout particulièrement dans les environnement urbains et péri-urbains.

La capacité de ces véhicules à circuler sans intervention humaine dans des environnements complexes représente un défi technologique majeur. On définit plusieurs degrés d'autonomie des véhicules. Ainsi une voiture autonome de niveau 4 est un véhicule capable de se déplacer seul dans un environnement défini comme une ville ou une autoroute. Le conducteur n'a pas besoin de garder les mains sur le volant mais il doit être capable de reprendre le contrôle du véhicule si nécessaire.



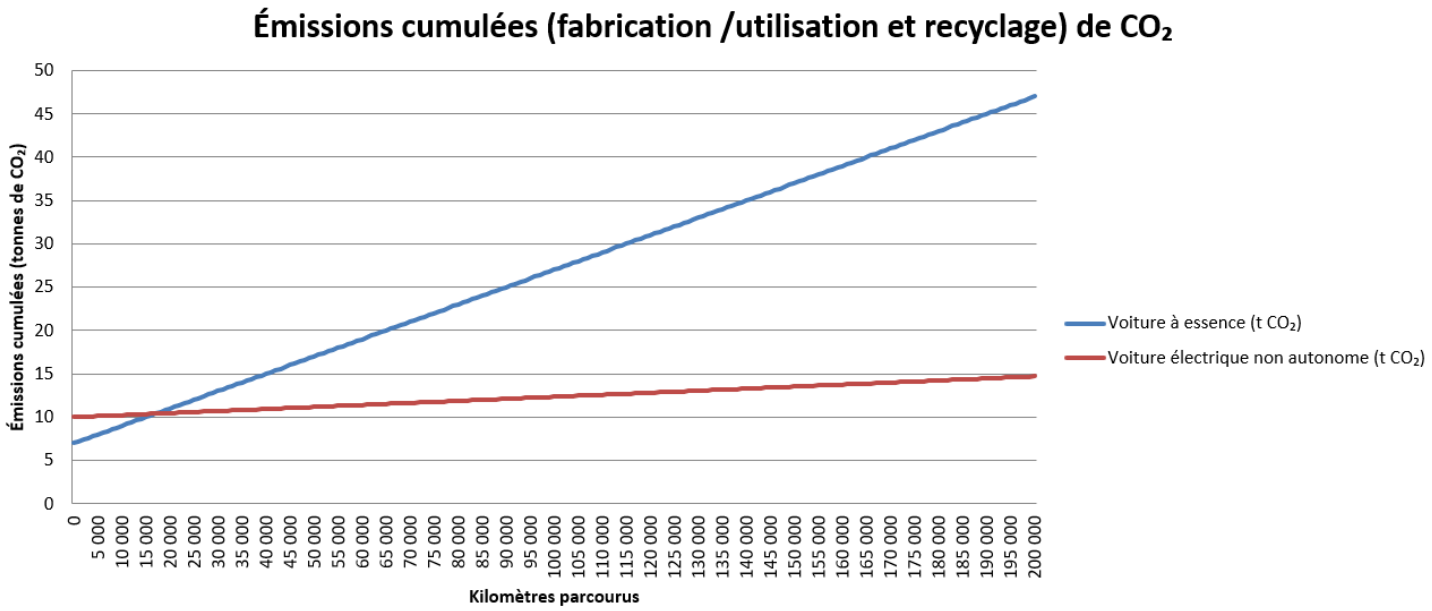
Partie Étude de l'objet technique

Usages et évolutions technologiques

Question 1. Par comparaison avec un véhicule à essence disposant de dispositifs d'aide à la conduite, citer deux évolutions technologiques nécessaires à l'apparition de véhicules autonomes de niveau 4.

Question 2. Indiquer les avantages/inconvénients de disposer de véhicules autonomes de niveau 4 pour un conducteur et pour la société dans son ensemble.

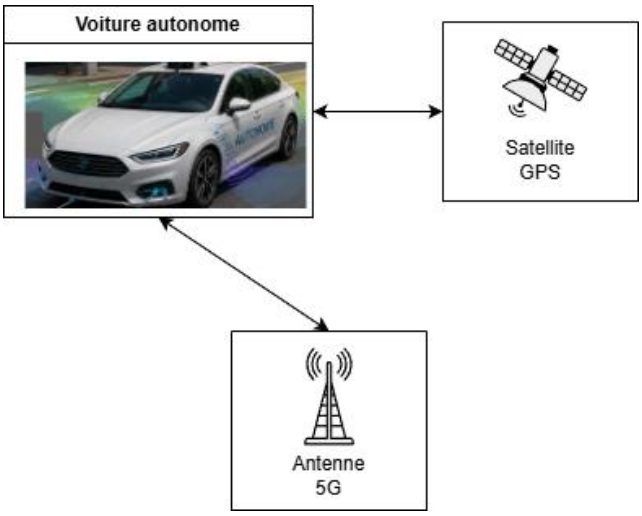
Le contexte du dérèglement climatique invite à questionner les conséquences environnementales des technologies utilisées dans le quotidien. La question de l'impact environnemental de véhicules autonomes se pose au regard des autres technologies de voitures actuellement en circulation.



Question 3. À l'aide du graphique ci-dessus, comparer les émissions de CO_2 entre un véhicule à essence et un véhicule électrique non autonome pour chaque étape du cycle de vie (fabrication, utilisation et recyclage). Indiquer le véhicule dont l'impact environnement global est le plus réduit. Indiquer comment évoluent les émissions de CO_2 entre un véhicule électrique non autonome et un véhicule électrique autonome de niveau 4, justifier la réponse.

Environnement et structure interne du véhicule autonome

Question 4. Compléter le diagramme suivant en plaçant au moins trois autres interacteurs autour de la voiture autonome. Pour chaque interacteur présent sur le diagramme, indiquer s'il s'agit d'usagers, de données, d'objets ou d'éléments de l'environnement.



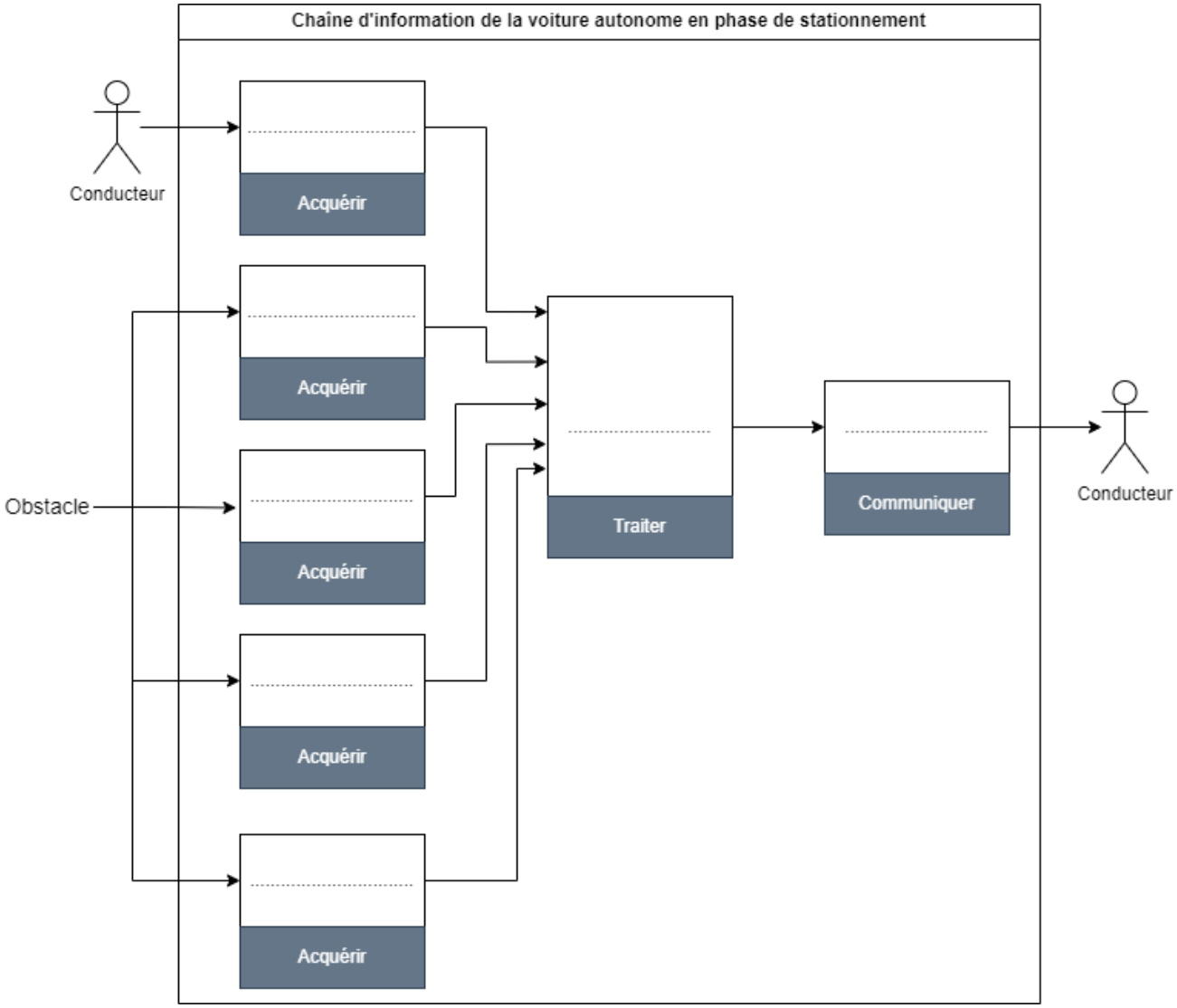
Afin de permettre à la voiture d'évoluer dans un environnement complexe, il est nécessaire de la doter de capteurs. Le tableau ci-dessous regroupe trois technologies de capteur permettant de mesurer des distances autour du véhicule.

Capteur	**Principe**	**Distance de détection**	**Précision**	**Conditions météorologiques**
Caméra	Capture d'images Traitement informatique pour la détection des formes	Variable selon objectif de la caméra et luminosité extérieure	Haute notamment pour la détection de la signalisation routière	Perte de précision en cas de faible luminosité extérieure
Radar	Émission d'ondes radio qui sont réfléchies par les obstacles Mesure du temps entre l'émission et l'onde de retour	Jusqu'à 250 m	Bonne	Reste efficace en cas de pluie ou de brouillard
Lidar	Émission d'impulsions lasers Mesure du temps entre l'émission et l'onde de retour	Entre 50 m et 250 m	Très haute	Perte d'efficacité dans les environnements avec de la neige et/ou de la poussière

Ultrasons	Émission d'ondes radio qui sont réfléchies par les obstacles Mesure du temps entre l'émission et l'onde de retour	Quelques mètres	Élevée	Reste efficace dans les environnements poussiéreux. La précision peut être affectée par la température ou l'humidité.

Question 5. À l'aide du tableau ci-dessus, justifier que les véhicules autonomes de niveau 4 soient équipés simultanément des quatre technologies de capteurs.

Question 6. À l'aide de l'illustration de la page 1, compléter la chaine d'information du véhicule autonome de niveau 4 lui permettant de percevoir son environnement direct et d'interagir avec le conducteur.



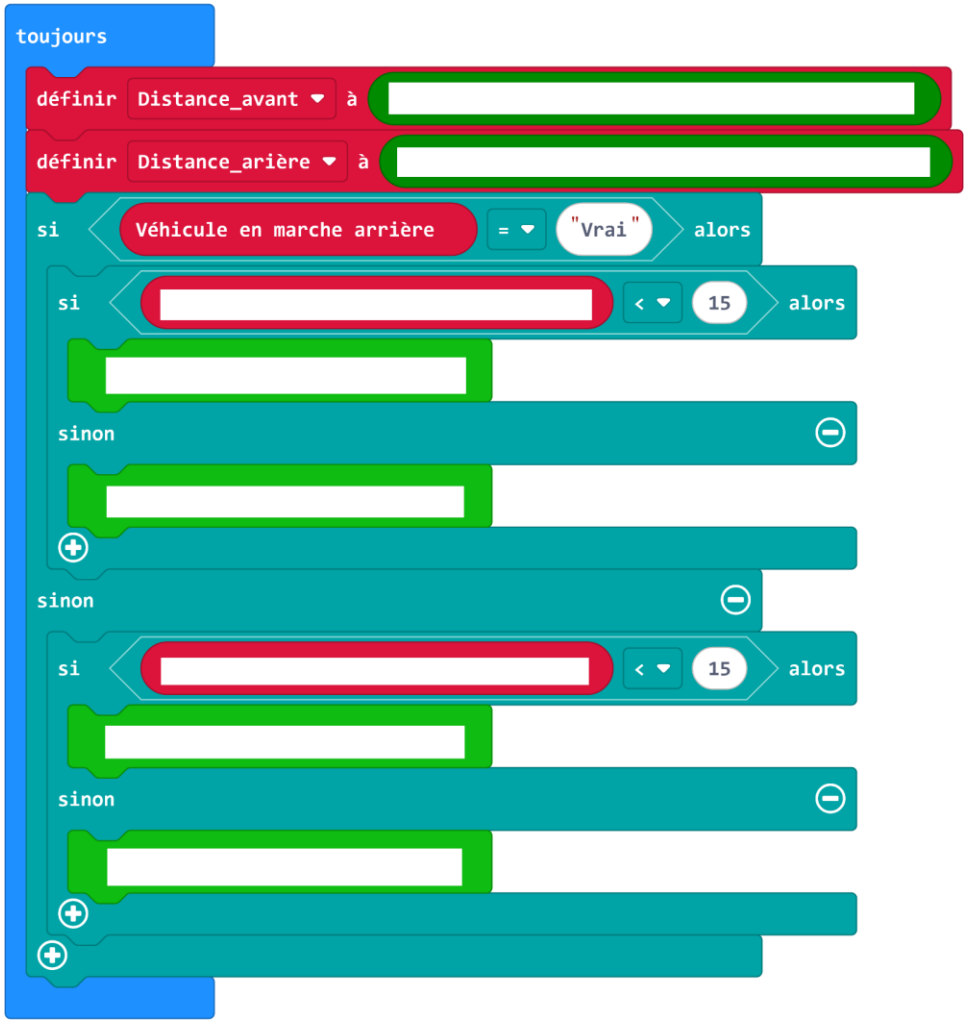
Informatique embarquée

Un véhicule autonome de niveau 4 intègre de multiples fonctionnalités comme l'aide au parking. Lors de la manœuvre la voiture mesure en permanence la distance avec les obstacles environnants. L'écran affiche une vue à 360° de l'environnement pour que le conducteur puisse contrôler le bon déroulement de la manœuvre.

Pour permettre à la voiture de mesurer les distances entre elle et les obstacles proches, des capteurs à ultrasons sont installés à l'avant et à l'arrière du véhicule. Dans une logique de simplification, on considère que le véhicule n'utilise que deux capteurs à ultrasons, un à l'avant (branché sur le port 3 de l'unité de traitement) et un à l'arrière (branché sur le port 4 de l'unité de traitement).

Lors de la phase de stationnement automatique, la voiture ne doit pas se rapprocher à moins de 15 cm d'un obstacle. Les capteurs à ultrasons renvoient la distance à l'obstacle le plus proche sous la forme d'un entier correspondant à la distance en cm (ex : une valeur de 24 renvoyée par un capteur à ultrason indique que l'obstacle le plus proche se trouve à 24 cm).

Question 7. Compléter le programme par blocs ci-dessous avec les propositions : *Avancer*, *Distance mesurée par le capteur à ultrasons sur le port 3*, *Reculer*, *Distance mesurée par le capteur à ultrasons sur le port 4*, *Arrêter*.



Au regard de l'autonomie des véhicules électriques actuels, la réalisation d'un voyage de longue distance avec un véhicule électrique autonome nécessite de prévoir des arrêts aux bornes de recharge. Afin d'éviter toute panne d'énergie de la voiture autonome, cette dernière dispose de plusieurs informations en temps réel :

- l'état de charge de sa batterie,
- la consommation énergétique estimée pour parcourir 100 km (en se basant sur les données du parcours réalisé précédemment),
- les distances (sur le trajet prévu) des trois stations de recharge les plus proches (déterminées grâce au GPS et aux données disponibles sur Internet).

La voiture autonome étudiée a une batterie ayant une capacité de 75 kWh. L'évaluation de la distance pouvant être encore parcourue par la voiture est déterminée grâce à la fonction `estimation_distance_restante` suivante :

```
fonction estimation_distance_restante(Batt, Conso)
    // Batt : pourcentage de charge de la batterie
    // Conso : consommation énergétique pour 100 km
    Capacite <- 75
    Energie_disponible <- Capacite × Batt / 100
    Distance <- (Energie_disponible / Conso) × 100
    renvoyer Distance
```

Cette fonction prend en paramètres le pourcentage de charge de la batterie (`Batt`), la consommation énergétique estimée pour parcourir 100 km (`Conso`) et renvoie la distance maximale que peut parcourir le véhicule en consommant l'intégralité de l'énergie disponible dans la batterie du véhicule. Dans le cas d'une voiture ayant une batterie chargée à 100% et d'une consommation énergétique estimée de 25 kWh pour 100 km, la distance maximale pouvant être parcourue sera obtenue grâce à l'appel `estimation_distance_restante(100, 25)`.

Afin de préserver le fonctionnement de la batterie tout au long de la vie du véhicule, il est nécessaire de s'assure que son état de charge ne soit jamais inférieur à 10%.

Question 8. Modifier la fonction `estimation_distance_restante(Batt, Conso)` afin qu'elle renvoie la distance maximale que peut parcourir le véhicule sans endommager la batterie.

Lors d'un trajet, l'écran permet d'informer le conducteur, au fur et à mesure du trajet, du besoin (ou non) de prévoir un arrêt en station de recharge. Le véhicule met à jour en temps réel les distances entre la position actuelle du véhicule et des trois prochaines stations de recharge présentes sur le trajet. La variable `D1` (respectivement `D2` et `D3`) contient la distance en km jusqu'à la 1ère station de recharge (respectivement la 2e et la 3e). On suppose `D1` < `D2` < `D3`.

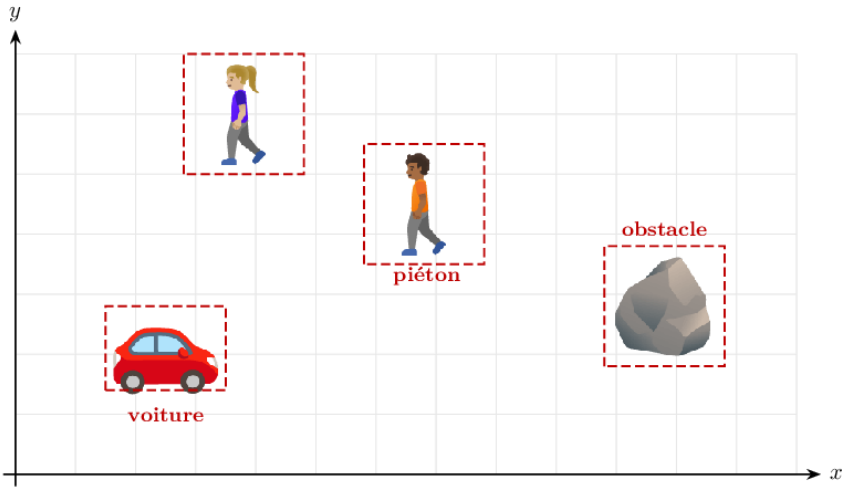
Question 9. Écrire en pseudocode une fonction `affichage(Batt, Conso, D1, D2, D3)` qui permet d'afficher sur l'écran du conducteur l'un des messages suivants selon la situation :

- « Batterie insuffisante pour atteindre la station de recharge la plus proche »
- « Prévoir un arrêt à la station de recharge X » (où X vaut 1, 2 ou 3 selon la station de recharge la plus adaptée)
- « Pas de recharge à prévoir dans l'immédiat »

Partie Algorithmique - Système de détection de collision

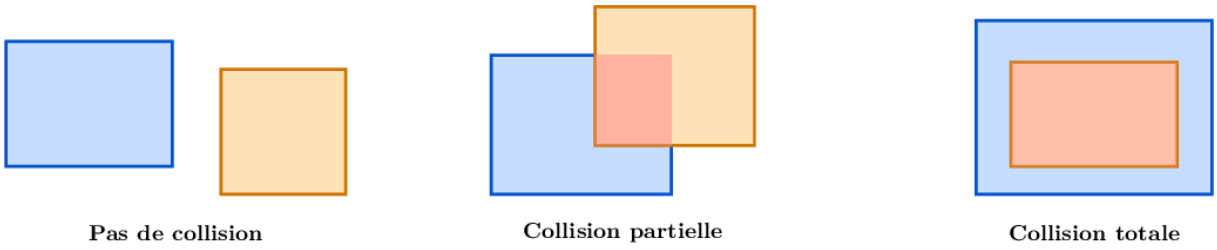
Présentation du problème

Dans cette partie, nous allons réaliser un système de détection de collision rudimentaire. Pour cela, on assimile véhicules et obstacles à des rectangles alignés sur les axes d'un repère orthogonal du plan, c'est-à-dire des rectangles dont les côtés sont tous horizontaux ou verticaux. On ne considère que deux dimensions, cela revient à voir les véhicules et obstacles **du dessus**.



Différents éléments de la scène et les rectangles qui les représentent.

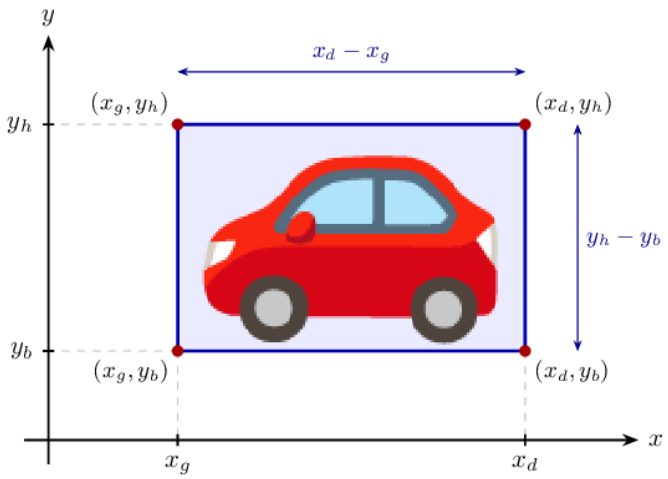
Le but de cette partie est d'être capable de décider quand deux rectangles sont en collision, autrement dit lorsqu'ils se superposent **partiellement ou totalement**.



Les trois situations possibles entre deux rectangles (la zone rouge indique la superposition).

Représentation des voitures et des piétons

On représente une voiture par un rectangle défini par quatre nombres `(xg, yb, xd, yh)` : `xg` est l'abscisse du bord gauche, `yb` l'ordonnée du bord bas, `xd` l'abscisse du bord droit et `yh` l'ordonnée du bord haut.

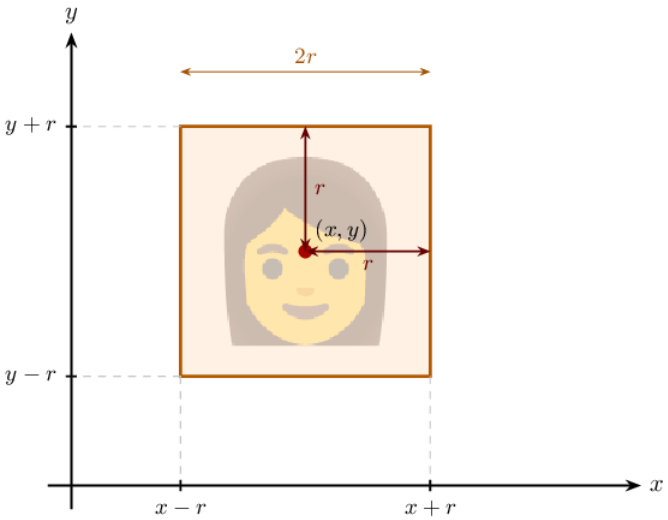


Représentation d'une voiture par le quadruplet (x_g, y_b, x_d, y_h).

Question 10. Montrer que l'aire d'une voiture dont $x_g = 1$, $y_b = 2$, $x_d = 4$ et $y_h = 7$ est égale à 15.

Question 11. Écrire **en pseudocode** une fonction `aire_voiture(xg, yb, xd, yh)` qui renvoie l'aire de la voiture représentée par `(xg, yb, xd, yh)`.

Les piétons sont représentés par des carrés définis par trois nombres `(x, y, r)` : `(x, y)` est le centre et `r` la distance du centre à chaque côté.



Représentation d'un piéton par le triplet (x, y, r).

Question 12. Montrer que le carré $x = 3$, $y = 5$, $r = 2$ est représenté par le quadruplet $(1,3,5,7)$.

Question 13. Écrire une fonction en pseudocode `rectangle_pieton(x, y, r)` qui renvoie les quatre nombres `(xg, yb, xd, yh)` décrivant le piéton.

Question 14. Écrire une fonction en pseudocode `point_dans_rectangle(xg, yb, xd, yh, x, y)` qui renvoie `VRAI` lorsque le point `(x, y)` est dans le rectangle `(xg, yb, xd, yh)` et `FAUX` sinon.

Tests de collisions

Avant de pouvoir réaliser un test de collision dans le plan, on s'intéresse au problème sur une droite graduée : on considère des *segments de nombres* comme `[xmin, xmax]` qui contiennent tous les nombres `x` compris entre deux nombres `xmin` et `xmax`, c'est -à-dire vérifiant `xmin <= x <= xmax`.

Le nombre 2 est présent dans les deux segments [1,5] et [3,7]. On dit que ces segments sont en *collision*.

Question 15. Écrire une fonction en pseudocode `collision_segment(xmin1, xmax1, xmin2, xmax2)` qui renvoie `VRAI` si les deux segments partagent un même nombre, et `FAUX` sinon.

Question 16. À l'aide de la fonction précédente, écrire une fonction en pseudocode `collision_rectangle(xg1, yb1, xd1, yh1, xg2, yb2, xd2, yh2)` qui renvoie `VRAI` si les deux rectangles partagent un même point, `FAUX` sinon.

En pseudocode, on peut manipuler des **listes** de valeurs. Si `L` est une liste, `longueur(L)` donne son nombre d'éléments et `L[i]` ($1 \leq i \leq$ longueur(L)) désigne la valeur de la liste à la position i, compris entre 1 et sa longueur.

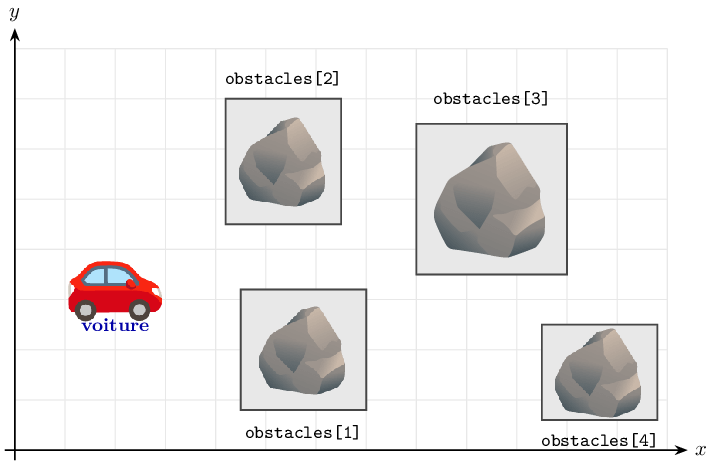
Par exemple, la fonction `somme_liste` prend en entrée une liste `L` de nombres et renvoie la somme de ses éléments.

```
fonction somme_liste(L)
  s <- 0
  pour i allant de 1 à longueur(L) faire
    s <- s + L[i]
  renvoyer s
```

Si `L` vaut [3,7,2,8], alors `somme_liste(L)` renvoie `20`.

On dispose d'une liste `obstacles` dont chaque élément est un rectangle. On récupère ainsi l'obstacle en position `i` en écrivant :

```
(xg, yb, xd, yh) <- obstacles[i]
```

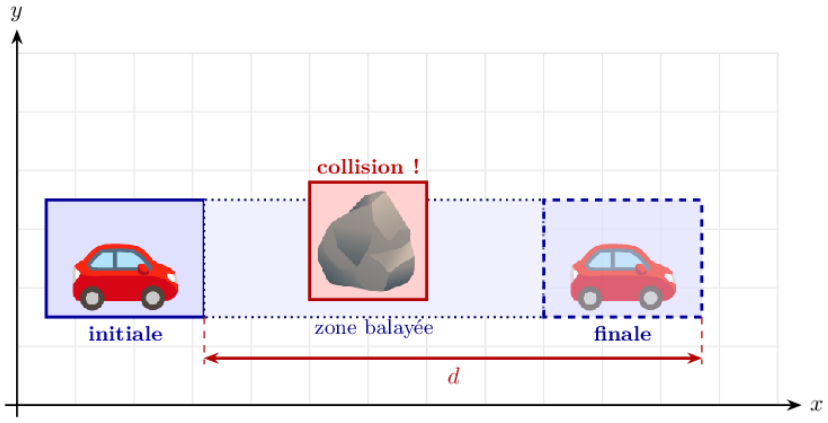


Une voiture et une liste de quatre obstacles ; la voiture n'est en collision avec aucun d'eux.

Question 17. Écrire une fonction en pseudocode `collision_obstacles(xgv, ybv, xdv, yhv, obstacles)` qui renvoie `VRAI` si la voiture est en collision avec **au moins un** obstacle, `FAUX` sinon.

Déplacement sûr

Une voiture en position `(xg, yb, xd, yh)` avance vers la droite d'une distance `d > 0`. Sa nouvelle position est `(xg + d, yb, xd + d, yh)`. Le déplacement est *sûr* si la voiture ne heurte aucun obstacle **pendant** son trajet.



Déplacement vers la droite sur une distance d ;
l'obstacle en rouge sera heurté pendant le trajet.

Question 18. Écrire une fonction en pseudocode `deplacement_sur(xg, yb, xd, yh, d, obstacles)` qui renvoie `VRAI` si la voiture peut se déplacer sans collision, `FAUX` sinon.