

UNIVERSITÉ DE SHERBROOKE
Département d'informatique

IFT 615
Intelligence artificielle

Examen final
Été 2009

Le mercredi 12 août, 9 h à 12 h, au D3-2030

CHARGÉ DE COURS SUPPLÉANT

Éric Beaudry

<http://planiart.usherbrooke.ca/~eric/>

INSTRUCTIONS

L'examen dure trois heures.

Les notes du cours (copie des présentations), le manuel (livres de référence) et les calculatrices sont autorisés. **Tout autre appareil électronique est strictement interdit, en particulier tout appareil muni d'un moyen de communication.**

L'examen comporte six questions pour un total de quarante points. Le questionnaire contient 11 pages incluant celle-ci.

Répondez directement sur le questionnaire aux endroits encadrés.

Des feuilles de brouillon vous sont fournies.

Ne détachez aucune feuille de ce questionnaire.

Écrivez votre nom, prénom et matricule ci-dessous.

NOM : _____ **PRÉNOM :** _____

MATRICULE : _____


SIGNATURE : _____

SOLUTIONS

Q1 /7	Q2 /9	Q3 /5	Q4 /5	Q5 /10	Q6 /4	TOTAL /40

Question 1 (7 points) – Processus décisionnels de Markov (MDP)

Un robot doit naviguer dans un environnement décomposé en une grille de neuf cellules étiquetées de c_1 à c_9 . La cellule c_3 est occupée par un obstacle et ne peut-être atteinte. Un état du robot est représenté par une paire (p, o) où p et o représentent respectivement la position (cellule) et l'orientation (haut, bas, gauche, droite) du robot. La figure ci-droite montre un robot dans l'état $(c_1, droite)$.

c_1 	c_2 [-50]	c_3
c_4	c_5	c_6 [+10]
c_7	c_8	c_9 [+20]

Le robot peut exécuter trois actions.

- a₁) Effectuer une rotation de 90° vers la gauche;
- a₂) Effectuer une rotation de 90° vers la droite;
- a₃) Avancer d'une case.

Les actions de rotation sont déterministes. Par contre, l'action d'avancer d'une case est incertaine. Suite à l'exécution de l'action a_3 , le robot peut ne pas avoir bougé ($P=5\%$), peut avoir avancé d'une seule case ($P=85\%$), ou peut avoir avancé de deux cases ($P=10\%$).

a) (1 point) Quelle est la taille de l'espace d'états accessibles?

8 cases accessibles * 4 états / case = 32 états accessibles.

b) (2 points) Simulez deux itérations de l'algorithme d'itération par valeurs sur les états $(c_1, droite)$ et $(c_9, haut)$. Les valeurs de tous les états sont initialisées à 2. Un facteur d'atténuation (*discount factor*) de 0.8 est utilisé. Les cellules c_2 , c_6 et c_9 contiennent des récompenses de -50, +10 et +20.

À la première itération, tous les états auront une valeur de $0.8 \times 2 = 1.6$, sauf $V[(c_2, .)] = 0.8 \times 2 - 50 = -48.4$, $V[(c_6, .)] = 10 + 0.8 \times 2 = 11.6$ et $V[(c_9, .)] = 20 + 0.8 \times 2 = 21.6$ en raison des récompenses.

Deuxième itération :

Valeur $[(c_1, droite)]$ = l'exécution de a_3 a comme successeur c_1 et c^2 . Donc le max sera obtenu à partir d'une rotation gauche ou droite pour atteindre les états $(c_1, haut)$ et (c_1, bas) qui ont une valeur de 1.6. Donc Valeur $[(c_1, droite)] = 0 + 0.8 \times (100\% \times 1.6) = 1.28$.

Valeur $[(c_9, haut)]$ = l'exécution d'une rotation est encore favorable pour éviter de sortir de la case c_9 qui a une valeur plus élevée que c_6 . Donc le max est obtenu à partir d'une action rotation. Donc Valeur $[(c_9, droite)] = 20 + 0.8 \times (100\% \times 21.6) = 37.28$.

c) (2 points) Vrai ou faux : la politique optimale est unique. Justifiez votre réponse avec un exemple.

Faux. De façon générale dans les MDP, il peut exister plusieurs politiques optimales ayant la même valeur.

Par exemple, si on est l'état $(c_9, droite)$, effectuer une rotation à gauche atteindra l'état $(c_9, haut)$ et une rotation à droite atteindra (c_9, bas) . Comme les états $(c_9, haut)$ et (c_9, bas) convergeront vers la même valeur numérique ($20 + 20 \times 0.8 = 36$), alors ces deux actions sont équivalentes dans l'état $(c_9, droite)$. Donc, il existe plusieurs politiques optimales.

d) (2 points) Vrai ou faux : l'exécution d'une politique optimale ne déplacera jamais le robot sur c_2 .

Vrai. L'action de rotation est déterministe et ne peut changer le robot de cellule. Ainsi, les actions rotations seront sélectionnées puisqu'elles donneront des valeurs plus grandes que les actions qui pourraient déplacer le robot sur c_2 qui a une valeur fortement négative.

Question 2 (9 points) – Logique du premier ordre

Un policier sollicite votre aide pour mener une enquête afin de déterminer si un conducteur intercepté sur la route est en légalité. Pour ce faire, vous disposez des règles suivantes :

1. Pour être légal sur la route, tout conducteur doit posséder un permis de conduire valide et avoir une assurance automobile.
2. Toute personne propriétaire d'une voiture est un conducteur.
3. La compagnie Yenma offre gratuitement une assurance automobile à tous ses employés.
4. Tout programmeur est un employé.
5. Toutes les Honda Civic sont des voitures économiques.

Après communication avec les services de renseignements, les faits suivants sont réputés vrais :

6. Jean possède une Honda Civic et un permis de conduire valide.
7. Jean est une personne qui occupe un poste de programmeur chez Yenma.

a. (3 points) Exprimez les règles et les faits précédents en logique du premier ordre.

1. $\forall x \text{ Conducteur}(x) \wedge \text{Permis}(x) \wedge \text{Assurance}(x) \rightarrow \text{Légal}(x)$
2. $\forall x \forall y \text{ Voiture}(y) \wedge \text{Possède}(x, y) \rightarrow \text{Conducteur}(x)$
3. $\forall x \text{ Employé}(x, \text{Yenma}) \rightarrow \text{Assurance}(x)$
4. $\forall x \forall y \text{ Programmeur}(x, y) \rightarrow \text{Employé}(x, y)$
5. $\forall x \text{ HondaCivic}(x) \rightarrow \text{Voiture}(x) \wedge \text{Economique}(x)$
6. $(\exists x (\text{Possède}(\text{Jean}, x) \wedge \text{HondaCivic}(x))) \wedge \text{Permis}(\text{Jean})$
7. $\text{Programmeur}(\text{Jean}, \text{Yenma})$

* Plusieurs réponses possibles.

b. (3 points) Convertissez les formules de l'étape précédente sous forme normale conjonctive. Numérotez chacune des clauses pour pouvoir y faire référence dans la sous-question suivante.

1. $\neg \text{Conducteur}(x1) \vee \neg \text{Permis}(x1) \vee \neg \text{Assurance}(x1) \vee \text{Légal}(x1)$
2. $\neg \text{Voiture}(x4) \vee \neg \text{Possède}(x3, x4) \vee \text{Conducteur}(x3)$
3. $\neg \text{Employé}(x5, \text{Yenma}) \vee \text{Assurance}(x5)$
4. $\neg \text{Programmeur}(x6, x7) \vee \text{Employé}(x6, x7)$
- 5a. $\neg \text{HondaCivic}(x8) \vee \text{Voiture}(x8)$
- 5b. $\neg \text{HondaCivic}(x9) \vee \text{Economique}(x9)$
- 6a. $\text{HondaCivic}(f(\text{Jean}))$
- 6b. $\text{Possède}(\text{Jean}, f(\text{Jean}))$
- 6c. $\text{Permis}(\text{Jean})$
7. $\text{Programmeur}(\text{Jean}, \text{Yenma})$

c. (3 points) Utilisez la preuve par résolution afin de déterminer si Jean est légal ou non.

Il faut nier Légal(Jean) :

8. $\neg \text{Légal}(\text{Jean})$

9. $\neg \text{Conducteur}(\text{Jean}) \vee \neg \text{Permis}(\text{Jean}) \vee \neg \text{Assurance}(\text{Jean})$ // 1+8, Upg : {x1=Jean}

10. $\neg \text{Conducteur}(\text{Jean}) \vee \neg \text{Assurance}(\text{Jean})$ // 9+6c Upg {}

11. $\text{Voiture}(f(\text{Jean}))$ // 5a + 6a {x8=f(Jean)}

12. $\neg \text{Voiture}(f(\text{Jean})) \vee \text{Conducteur}(\text{Jean})$ // 2+6b {x3=Jean, x4=f(Jean)}

13. $\text{Conducteur}(\text{Jean})$ // 12+11

14. $\neg \text{Assurance}(\text{Jean})$ // 10+13

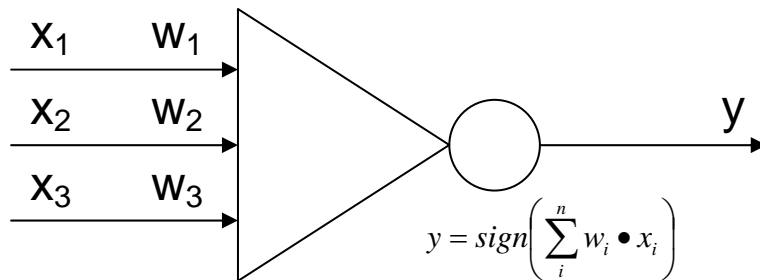
15. $\text{Employé}(\text{Jean}, \text{Yenma})$ // 4+7 {x6=Jean, x7=Yenma}

16. $\text{Assurance}(\text{Jean})$ // 3+15 {x5=Jean}

17. FAUX // 14+16

Question 3 (5 points) – Réseau de neurones

Soit le neurone artificiel suivant.



Les deux données suivantes sont utilisées pour entraîner le neurone.

x_1	x_2	x_3	y
4	2	0	1
2	4	7	1

a) (3 points) Simulez une itération complète de l'algorithme d'apprentissage du perceptron en utilisant les deux données d'entraînement. Le pas d'apprentissage est fixé à 0.2 et les poids initiaux sont $w_1=1$, $w_2=2$ et $w_3=0$.

Première donnée :

Sortie = $\text{sign}(1*4 + 2*2 + 0*0) = 1$

$\Delta w = 0.2 * (1-1)*X = 0$ // inchangé

Deuxième donnée :

Sortie = $\text{sign}(2*1 + 2*4 + 7*0) = 1$

$\Delta w = 0.2 * (1-1)*X = 0$ // inchangé

Aucun changements.

$w_1=1$, $w_2=2$ et $w_3=0$

b) (2 points) Vrai ou faux : le neurone ci-haut peut apprendre parfaitement tout jeu d'entraînement de dix (10) exemples, à la condition qu'on lui donne un nombre suffisant d'itérations. Justifiez votre réponse.

FAUX. Pour qu'un neurone puisse apprendre parfaitement un jeu de données, ces dernières doivent être linéairement séparable. Avec 10 entrées, il n'est pas garanti qu'on puisse trouver un plan (défini par une équation) qui sépare ces 10 données.

Question 4 (5 points) – Réseau de neurones appliqué

a) (3 points) Vous avez pour mission d'écrire un programme capable de reconnaître des mots écrits à l'aide de lettres moulées. Votre client exige que votre solution soit basée sur un réseau de neurones (RN). Détaillez les entrées et sortie(s) de votre RN (type de données, nombres d'entrées et sortie(s)). Utilisez une illustration pour montrer la structure du réseau.

Méthode : on isole chaque lettre.

Entrées : matrice de pixels, il y a autant de neurones que de pixels.

Sorties : 26 sorties qui donne le niveau de reconnaissance d'une lettre a à z.

[Mettre un dessin ici]

b) (2 points) Des premiers tests montrent que votre reconnaît correctement les lettres dans 76 % des cas et les mots dans 37 % des cas. Comment pourriez-vous améliorer ces résultats en intégrant une autre technique d'IA à votre programme?

Une façon simple consisterait à utiliser un dictionnaire pour favoriser le mot le plus probable.

On peut aussi utiliser des chaînes cachées de Markov.

Question 5 (10 points) – Planification

Un employé doit imprimer un important rapport sauvegardé sur sa clé USB. L'édifice dans lequel il se trouve ne contient aucun ordinateur ayant un port USB et relié à une imprimante. La compagnie a fait l'acquisition d'un planificateur pour aider ses employés à utiliser les ressources informatiques. La spécification PDDL suivante décrit une partie du domaine de planification pour l'impression. Une variable débute par le symbole « ? ».

```
(define (domain Impression)
  (types PC Salle) ;; définit 2 types d'objets: PC pour ordinateur et salle
  (:predicates
    (ReliéAuRéseau ?pc - PC) ;; Le PC ?pc est relié au réseau
    (Dans ?s - Salle) ;; L'employé est dans la salle ?s
    (Contient ?s - Salle ?pc - PC) ;; La salle ?s contient le PC ?pc
    (ReliéImprimante ?pc - PC) ;; Le PC ?pc a une imprimante
    (PortUSB ?pc - PC) ;; Le PC ?pc dispose d'un port USB
    (FichierPresent ?pc - PC) ;; Le fichier est sur le PC ?pc
    (RapportImprimé) ;; Le rapport est imprimé
  )

  (:action CopierFichierDepuisCléUSB
    :parameters(?pc - PC ?salle - Salle)
    :preconditions(and (PortUSB ?pc) (Dans ?salle) (Contient ?salle ?pc))
    :effects( (FichierPresent ?pc) )
  )

  (:action Marcher
    :parameters(?origine - Salle ?destination - Salle)
    :preconditions(Dans ?origine)
    :effects(and (not (Dans ?origine)) (Dans ?destination))
  )

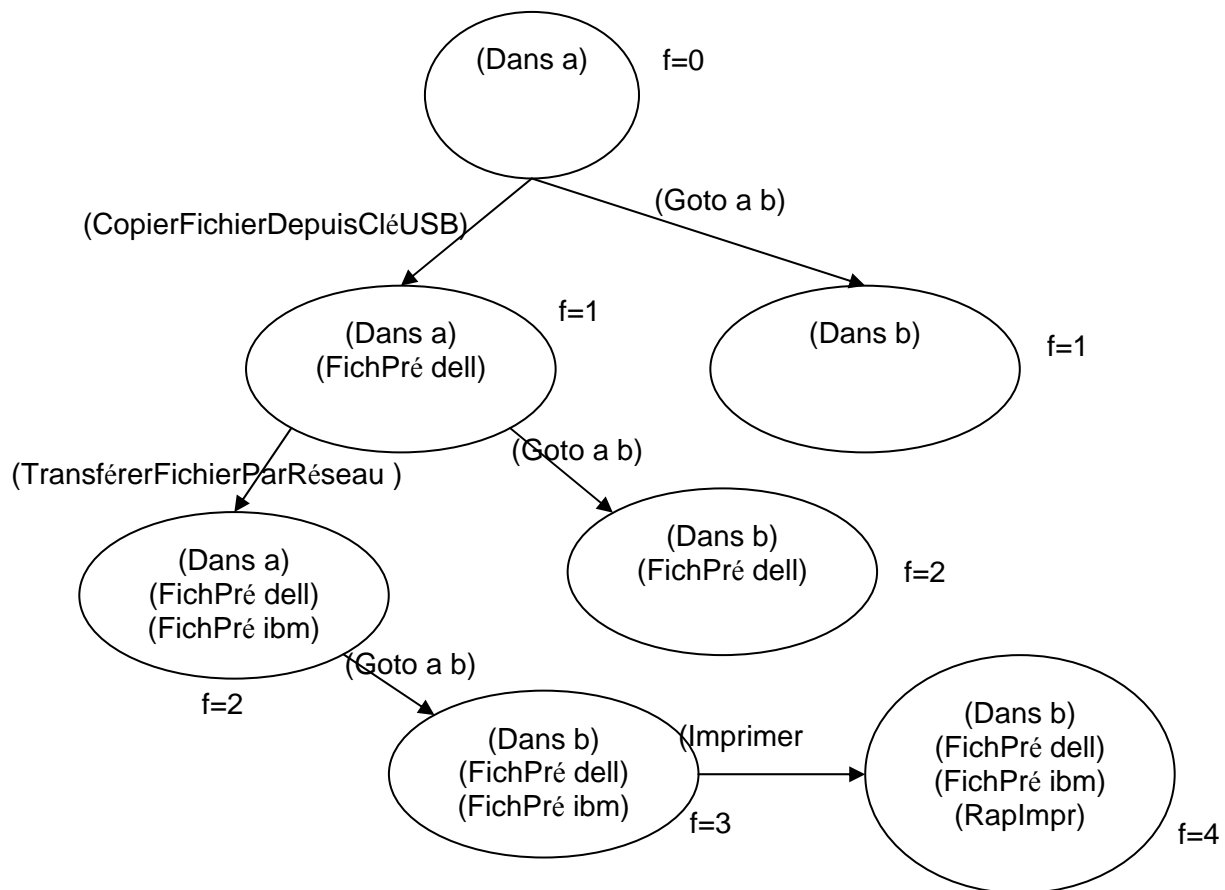
  (:action TransférerFichierParRéseau
    :parameters(?pc1 - PC ?salle - Salle ?pc2 - PC )
    :preconditions(and (FichierPresent ?pc1) (Contient ?salle ?pc1)
      (Dans ?salle) (ReliéAuRéseau ?pc1) (ReliéAuRéseau ?pc2) )
    :effects( (FichierPresent ?pc2) )
  )
)
```

a) (2 points) Donnez le code PDDL pour l'action manquante *ImprimerFichier*.

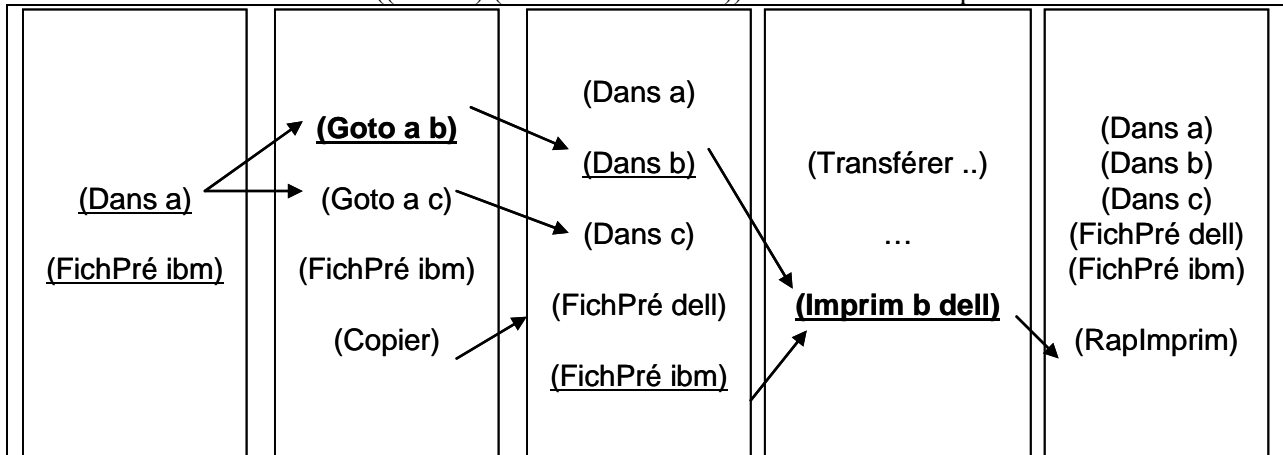
```
(:action ImprimerFichier
  :parameters(?pc - PC ?salle - Salle)
  :preconditions(and (FichierPresent ?pc) (Contient ?salle ?pc)
    (Dans ?salle) (ReliéImprimante ?pc))
  :effects( (RapportImprimé) )
)
```

b) (3 points) Pour le fichier problème suivant, dessinez au moins quatre (4) états d'un graphe de recherche pouvant être généré par un planificateur utilisant l'algorithme A* (chaînage avant) piloté sans heuristique. Vous pouvez utiliser des abréviations pour le nom des actions et prédicats. Sur chaque état, notez la valeur de f obtenu à l'aide de la formule $f = g + h$.

```
define (problem p1)
  (:objects
    dell ibm toshiba - PC
    a b c - Salle)
  (:init (and
    (Contient a dell) (Contient b ibm) (Contient c toshiba)
    (ReliéAuRéseau dell) (ReliéAuRéseau ibm)
    (Dans a) (PortUSB dell) (ReliéImprimante ibm) ))
  (:goal (RapportImprimé)) )
```



c) (2 points) Quel est le nombre d'actions restantes estimé par l'heuristique « *Relaxed GraphPlan* » pour un état contenant les littéraux ((Dans a) (FichierPrésent ibm)) ? Justifiez votre réponse à l'aide d'un dessin.



Pour simplifier l'écriture, les actions no-op ne sont pas représentées.

Nombre estimé d'actions restantes : (Goto a b) (Imprimer b dell) : 2 actions.

d) (3 points) L'employé ne veut laisser aucune trace de son document sur les ordinateurs qu'il a utilisés. Donnez les modifications requises pour répondre à ce nouveau besoin.

Plusieurs options existent. La plus simple est d'ajouter une action pour effacer un fichier présent sur un PC et de mettre dans le goal (not (FichierPrésent pc)) pour chaque PC.

Il est possible d'utiliser le quantificateur existentiel pour tout pour spécifier tous les PC à la fois.

```
(:action SupprimerFichier
  :parameters(?pc - PC ?salle - Salle)
  :preconditions(and (FichierPrésent ?pc) (Contient ?salle ?pc) )
  :effects(not (FichierPrésent ?pc))
)

(:goal (and (RapportImprimé)
  (not (FichierPrésent dell))
  (not (FichierPrésent ibm))
  (not (FichierPrésent toshiba))
))
```

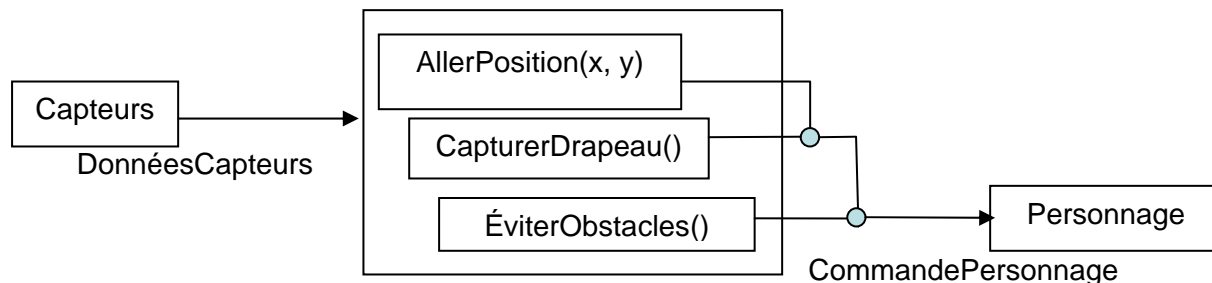
Une autre option serait de garder un compteur sur le nombre de PC qui contient le fichier. À chaque copie depuis la clé USB ou transfère de fichier, on incrémente de un, et à chaque suppression, on décrémente de un.

Question 6 (4 points) – Exposés sur la robotique mobile (Éric Beaudry) et sur l'IA dans les jeux vidéo (Khaled Belghith)

Dans un jeu vidéo où le but est de voler le drapeau de l'adversaire, plusieurs personnages non jouables (*Non-Player Characters*) sont contrôlés automatiquement par l'ordinateur. Pour animer ces personnages, vous vous inspirez de techniques utilisées en robotique mobile, dont l'utilisation de modules comportementaux. Vous disposez des classes suivantes.

<pre>class Personnage { public: double x, y; // position double o; // orientation vector<Comportement*> comportements; };</pre>	<pre>class DonnéesCapteurs { public: double distances[180]; bool drapeauDétecté; double drapeauAngleDétecté; double drapeauDistanceDét; };</pre>	<pre>class CommandePersonnage { public : double vitesse; // m/s double rotation; // radian/s bool prendreObjet; };</pre>
---	--	--

Au moment de la conception, l'équipe de développement a décidé que trois comportements devaient être programmés contrôler les personnages non jouables. Le schéma suivant montre l'organisation par priorité de ces comportements.



Avant de quitter pour les vacances, un de vos collègues a programmé l'évitement d'obstacle.

```

class Comportement {
    Personnage personnage;
};
class ÉviterObstacles : public Comportement {
    void calculerCommande(DonnéesCapteurs dc, CommandePersonnage cmd){
        double gauche = trouverValeurMax(dc.distances, 0, 60);
        double centre = trouverValeurMax(dc.distances, 60, 120);
        double droite = trouverValeurMax(dc.distances, 120, 180);
        double tout = trouverValeurMax(dc.distances, 0, 180);
        if(tout<1.0)
            cmd.vitesse = 0; cmd.rotation = -15;
        else if(centre < 1.0)
            cmd.rotation = sign(gauche-droite) * 15;
    }
};
class CaptuerDrapeau : public Comportement {
    void calculerCommande(DonnéesCapteurs dc, CommandePersonnage cmd){
        if(dc.drapeauDétecté) {
            cmd.vitesse = 2;
            cmd.rotation = dc.drapeauAngleDétecté;
            if(dc.drapeauDistanceDét < 0.2) cmd.prendreObjet=true;
        }
    }
};
```

a) (2 points) Implémenter le comportement AllerPosition qui indique un point (x_b , y_b) à atteindre.

```
class AllerPosition : Comportement {
    double butX, butY;
    AllerPosition(double x, double y) butX(x), butY(y) {}

    void calculerCommande(DonnéesCapteurs dc, CommandePersonnage cmd){

        dx = butX - personnage.x;
        dy = butY - personnage.y;

        if( sqrt(dx*dx + dy*dy) < seuil ){
            // Marquer but atteint
            cmd.vitesse = 0;
            cmd.rotation = 0;
        }else{

            angleCible = atan2(dx, dy);
            diffAngle = angleCible - personnage.o;

            if(diffAngle < - PI ) diffAngle += *2PI;
            if(diffAngle > PI ) diffAngle -= *2PI;

            cmd.rotation = diffAngle / constante; //
            cmd.vitesse = vitesseMax; // ou 2
        }
    }
    /* Les réponses peuvent varier. L'important est que la rotation soit proportionnelle à l'angle vers la cible
    et qu'on demande au robot d'avancer tant que la cible n'est pas atteinte. */
};
```

b) (2 points) Une fois le jeu programmé, l'équipe chargée des tests trouve que votre jeu est trop difficile. Le personnage non jouable gagne trop facilement la partie. Donnez deux astuces pouvant être utilisées pour rendre le personnage artificiel moins bon.

1) Dans le comportement CaptuerDrapeau, on peut choisir une vitesse inférieure à 2. En fait, la vitesse pourrait être proportionnelle à la situation du joueur adverse. Plus il est prêt d'accomplir son but, plus notre vitesse sera proche de notre vitesse maximale.

2) On peut paramétrer la distance à laquelle le joueur intelligent est capable de détecter le drapeau.

3) On peut ajouter un peu de random dans la rotation des comportements pour ralentir le robot.

4) Le comportement CapturerDrapeau peut introduire une pause entre le temps où le drapeau est détecté et le moment où on le capture.

5) Si le joueur artificiel a un algorithme de planification de chemin, on peut donner un poids plus fort à l'heuristique pour ne pas obtenir des chemins optimaux.

...