Sujet C

❑ **Exercice** : *Exercice A*

De nombreux travaux sont réalisés en Intelligence Artificielle pour construire un programme qui imite le raisonnement humain et soit capable de réussir le test de Turing, c'est-à-dire qu'il ne puisse pas être distingué d'un être humain dans une conversation à l'aveugle. Vous êtes chargé(e)s de vérifier la correction des réponses données par un tel programme lors des tests de bon fonctionnement. Dans le scénario de test considéré, le comportement attendu est le respect de la règle suivante : pour chaque question, le programme répondra par trois affirmations et parmi ces affirmations une seule sera correcte (et donc les deux autres fausses).
Nous noterons A_1 , A_2 et A_2 les propositions associées aux affirmations effectuées par le programme.

1. Représenter le comportement attendu sous la forme d'une formule du calcul des propositions qui dépend de A_1, A_2 et A_3.

Premier cas : Vous demandez au programme : « *Quels éléments doivent contenir les aliments que je dois consommer pour préserver ma santé ?* »
Il répond les affirmations suivantes :

A_1 : Consommez au moins des aliments qui contiennent des glucides mais pas de lipides !

A_2 : Si vous consommez des aliments qui contiennent des glucides, alors ne consommez pas d'aliments qui contiennent des lipides !

A_3 : Ne consommez aucun aliment qui contient des lipides

Nous noterons G, respectivement L, les variables propositionnelles qui correspondent au fait de consommer des aliments qui contiennent des glucides, respectivement des lipides.

2. Exprimer A_1, A_2 et A_3 sous la forme de formules du calul des propositions. Ces formules peuvent dépendre des variables G et L.
3. En utilisant le calcul des propositions (résolution avec les formules de De Morgan), déterminer ce que doivent contenir les aliments que vous devrez consommer pour préserver votre santé.

Deuxième cas : Vous demandez au programme : « *Quelles activités dois-je pratiquer si je veux préserver ma santé ?* »
Suite à une coupure de courant, la dernière information est interrompue.

A_1 : Si vous faites des activités sportives alors prenez du repos !

A_2 : Si vous ne faites pas d'activités intellectuelles, alors ne prenez pas de repos !

A_3 : Prenez du repos ou faites des activités ... !

Nous noterons S, I et R les variables propositionnelles qui correspondent au fait de faire des activités sportives, des activités intellectuelles, et de prendre du repos.

4. Exprimer A_1, A_2 et A_3 sous la forme de formules du calul des propositions. Ces formules peuvent dépendre des variables S, I et R. Pour A_3, on indiquera des pointillés pour signifier que la réponse a été interrompue.
5. En utilisant le calcul des propositions (résolution avec les tables de vérité), déterminer quelle(s) activités vous devez pratiquer pour préserver votre santé. L'information A_3 étant incomplète ou pourra envisager les deux cas suivants : elle se termine par "sportives" ou "intellectuelles" et réaliser la table de vérité dans chacun des cas.

❑ **Exercice** : *Exercice B*

Les fonctions demandées dans cet exercice doivent être écrites en OCaml et on *s'interdit* dans cet exercice l'utilisation des aspects impératifs du langage (boucles et références notamment). On dispose d'un sac à dos pouvant contenir un poids maximal noté P et de n objets ayant chacun un poids $(p_i)_{0\leqslant i\leqslant n-1}$ et une valeur $(v_i)_{0\leqslant i\leqslant n-1}$. On cherche à remplir le sac à dos de manière à maximiser la valeur totale des objets contenus dans le sac sans dépasser le poids maximal P. Par exemple, si on dispose des objets suivants :

— un objet de poids $p_0 = 4$ et de valeur $v_0 = 20$,

— un objet de poids $p_1 = 5$ et de valeur $v_1 = 28$,

— un objet de poids $p_2 = 6$ et de valeur $v_2 = 36$,

— un objet de poids $p_3 = 7$ et de valeur $v_3 = 50$,

et qu'on suppose que le poids maximal du sac est 10 alors un choix possible serait de prendre l'objet 3, aucun autre objet ne rentre alors dans le sac et la valeur du sac est de 50 avec un poids de 7. Une autre possibilité plus intéressante serait de choisir les objets 0 et 2, la valeur totale serait alors de 56 et le poids du sac de 10.

Dans toute la suite de l'exercice on supposera que les poids et les valeurs des objets ainsi que le poids maximal du sac sont des entiers (type `int`), et que donc un objet être représenté par un couple (poids, valeurs) `int*int` et une liste d'objets par le type `(int*int) list` de OCaml. Et on définit le type suivant afin de représenter un problème du sac à dos :

```
type pbsac =
  {
    pmax : int;
    objets : (int*int) list
  }
```

Ainsi, le problème donné en exemple ci-dessus correspond à

```
  let ex =
    {
      pmax = 10;
      objets = [(7, 35); (4, 18); (5, 20);]
    }
```

Le fichier contenant cette définition de type et l'exemple de problème peut-être télécharger à l'adresse : `https://fabricenativel.github.io/cpge-info/oraux/`

On propose de représenter un choix d'objets par une liste de booléens. Si le i-ème élément de la liste vaut `true` alors l'objet i est choisi, sinon l'objet i n'est pas choisi. Par exemple, pour les objets précédents, le choix de prendre uniquement l'objet 3 serait représenté par la liste `[false; false; false; true]` et le choix de prendre les objets 0 et 2 serait représenté par la liste `[true; false; true; false]`.

1. Déterminer la complexité d'une méthode résolvant le problème du sac à dos par *recherche exhaustive*, que peut-on en conclure ?
2. On considère la stratégie gloutonne suivante : on trie les objets par ordre décroissant de leur rapport valeur/poids et on les choisit dans cet ordre. Montrer à l'aide d'un contre exemple que cette stratégie ne permet pas toujours d'obtenir la valeur maximale.
3. La documentation de la fonction `List.sort : ('a -> 'a -> int) -> 'a list -> 'a list` de la librairie standard d'OCaml indique que cette fonction trie une liste d'éléments en utilisant l'ordre défini par la fonction de comparaison passée en paramètre. Cette dernière renvoie 0 lorsque les deux éléments sont égaux, un entier positif lorsque le premier élément est plus grand et un entier négatif sinon. Cette documentation indique aussi que l'algorithme utilisé est le *tri fusion*. En déduire une fonction `tri : (int*int) list -> (int*int) list` qui trie une liste d'objets par ordre décroissant de leur rapport valeur/poids.
4. Ecrire une fonction `glouton : pbsac -> int`, qui prend en arguments un problème du sac à dos et qui renvoie la valeur maximale que l'on peut obtenir en appliquant la stratégie gloutonne.
5. Donner la complexité de la foncton `glouton` en fonction du nombre d'objets n.

On propose de résoudre le problème du sac à dos par programmation dynamique. Pour tout $i \in [\![0; n]\!]$, on note $V(i, p)$ la valeur maximale que l'on peut obtenir avec les objets à partir de celui d'indice i et un poids maximal p. Par exemple s'il y a 5 objets (numérotés de 0 à 4), $v(2, 10)$ est le poids maximal atteint pour un sac de poids maximal 10 en ne considérant que les objets 2, 3 et 4.

6. Donner $V(i, 0)$ pour $i \in [\![0; n-1]\!]$ et $V(n, p)$ pour $p \in [\![0; P]\!]$.
7. Donner les relations de récurrence liant $v(i, p)$ à $v(i+1, p)$ et $v(i+1, p-p_i)$.
8. Écrire une fonction récursive `dynamique : pbsac -> int` qui prend en arguments un problème du sac à dos et qui renvoie la valeur maximale que l'on peut obtenir en appliquant la stratégie de programmation dynamique.