Partie I - Programmation en OCaml : sélection du (k+1)^{e} plus petit élément

Question 1 –

```
let rec longueur (li: 'a list): int = match li with
  | [] -> 0
  | _ :: q -> 1 + longueur q;;
```

Question 2 –

```
let rec insertion (li: 'a list) (a: 'a): 'a list = match li with
  | [] -> [a]
  | e :: q when a < e -> a :: li
  | e :: q -> e :: insertion q a;;
```

Question 3 –

```
let rec tri_insertion (li: 'a list): 'a list = match li with
  | [] -> []
  | e :: q -> insertion (tri_insertion q) e;;
```

Question 4 –

```
let rec selection_n (li: 'a list) (n: int): 'a = match li with
  | [] -> failwith "selection_n: pas d'élément de rang n"
  | e :: _ when n = 0 -> e
  | e :: q -> selection_n q (n-1);;
```

Question 5 –

```
let rec paquets_de_cinq (li: 'a list): 'a list list = match li with
  | [] -> []
  | e1 :: e2 :: e3 :: e4 :: e5 :: q -> [e1; e2; e3; e4; e5] :: paquets_de_cinq q
  | li -> [li];;
```

Question 6 –

```
let rec medians (li: 'a list list): 'a list = match li with
  | [] -> []
  | x :: q -> let n = longueur x in
              let x = tri_insertion x in
              selection_n x (n/2) :: medians q;;
```

Remarque. Il semble y avoir une erreur d'énoncé :

`medians [[3;1;5;3;2]; [4;3;1]; [1;3]; [5;1;2;4]]` est égal à `[3; 3; 3; 4]`.

Question 7 –

```
let rec partage (p: 'a) (li: 'a list): 'a list * 'a list * int * int =
  match li with
  | [] -> ([], [], 0, 0)
  | e :: q -> let (li1, li2, n1, n2) = partage p q in
              if e <= p then
                (e :: li1, li2, n1+1, n2)
              else
                (li1, e :: li2, n1, n2+1);;
```

Question 8 –

```
let rec selection (li: 'a list) (k: int): 'a = match longueur li with
  | n when n <= 5 -> selection_n (tri_insertion li) k
  | n -> let li_cinq = paquets_de_cinq li in
         let m = medians li_cinq in
         let pivot = selection m (((n+4)/5)/2) in
         let (li1, li2, n1, n2) = partage pivot li in
         if k < n1 then
           selection li1 k
         else
           selection li2 (k-n1);;
```

Question 9 – On le montre par récurrence sur $n \in \mathbb{N}^*$.

Initialisation. Si $n \in [\![1\,;55]\!]$. Comme la fonction T est croissante et positive, on a :

$$T(n) \leqslant T(55) \leqslant 200 + T(55) \leqslant (200 + T(55))n$$

Hérédité Soit $n \geqslant 56$. On suppose l'inégalité vraie pour tout entier $n' \in [\![1\,; n-1]\!]$ et on la montre pour n. D'après la proposition (I) :

$$T(n) \leqslant T\left(\left\lfloor \frac{n+4}{5} \right\rfloor\right) + T\left(\left\lfloor \frac{8n}{11} \right\rfloor\right) + 4n$$

De plus, pour tout $n \geqslant 56$:

$$n - 1 - \left\lfloor \frac{n+4}{5} \right\rfloor \geqslant n - 1 - \frac{n+4}{5} = \frac{4n-9}{5} \geqslant 0 \qquad \text{donc} \qquad \left\lfloor \frac{n+4}{5} \right\rfloor \leqslant n-1$$

De même :

$$\left\lfloor \frac{8n}{11} \right\rfloor \leqslant n - 1$$

On peut donc appliquer l'hypothèse de récurrence deux fois pour obtenir :

$$\begin{aligned} T(n) &\leqslant \Big(200 + T(55)\Big) \left\lfloor \frac{n+4}{5} \right\rfloor + \Big(200 + T(55)\Big) \left\lfloor \frac{8n}{11} \right\rfloor + 4n \\ &\leqslant \Big(200 + T(55)\Big) \left(\frac{n+4}{5} + \frac{8n}{11}\right) + 4n \\ &= \Big(200 + T(55)\Big) \frac{51n + 44}{55} + 4n \end{aligned}$$

Or :

$$44 \leqslant n \qquad \text{et} \qquad 4n \leqslant 200 \times \frac{2n}{55} \leqslant \Big(200 + T(55)\Big) \frac{2n}{55}$$

Ainsi :

$$\begin{aligned} T(n) &\leqslant \Big(200 + T(55)\Big)\frac{51n+n}{55} + \Big(200 + T(55)\Big)\frac{2n}{55} \\ &= \Big(200 + T(55)\Big)\frac{54n}{55} \\ &\leqslant \Big(200 + T(55)\Big)n \end{aligned}$$

Partie II - Recherche d'une clique de célébrités

Question 10 –

1.

Le graphe G_1 ne contient pas de clique de célébrités non vide.

En effet, si un sommet u correspond à une célébrité, alors toutes les autres personnes connaissent u. Étant donné qu'aucun sommet de G_1 n'est connu de tous les autres, il n'existe pas de clique de célébrités non vide dans G_1.

2.

L'ensemble $\{3\}$ est une clique de célébrités non vide dans le graphe G_2.

Question 11 – Soient C_1 et C_2 deux cliques de célébrités non vides dans G. Montrons que $C_1 = C_2$. Soit $c_1 \in C_1$ et $c_2 \in C_2$.

Comme C_1 est une clique de célébrités et que $(c_1, c_2) \in C_1 \times S$, on a :

$$(c_2, c_1) \in A$$

Comme C_2 est une clique de célébrités et que $(c_2, c_1) \in C_2 \times S$, on a :

$$(c_2, c_1) \in A \Rightarrow c_1 \in C_2$$

Des deux points précédents, on déduit que $c_1 \in C_2$. Finalement, $C_1 \subset C_2$ et de manière analogue $C_2 \subset C_1$, d'où le résultat.

Question 12 – Commençons par montrer la proposition b). Soit $C' = C_G \setminus \{p\}$. Supposons que $C' \neq \emptyset$ et montrons que $C_{G'} = C'$. Par unicité de $C_{G'}$, il suffit de montrer que C' est une clique de célébrités de G'. Vérifions que C' satisfait la définition 3 de l'énoncé. Soient :

$$S' = S \setminus \{p\} \qquad \text{et} \qquad A' = A \cap (S \setminus \{p\} \times S \setminus \{p\})$$

de sorte que $G' = (S', A')$. Alors :

→ C' est une clique de G' car C_G est une clique de G.

→ Pour tout $(c', s') \in C' \times S'$, on a $(c', s') \in C_G \times S$. Ainsi :

- Comme C_G est une clique de célébrités de G, on a $(s', c') \in A$. De plus, $c' \neq p$ et $s' \neq p$, donc $(s', c') \in A$.
- Supposons que $(c', s') \in A'$ et montrons que $s' \in C'$. Comme $(c', s') \in A' \subset A$ et que C_G est une clique de célébrités de G, on a $s' \in C_G$. Comme $s' \neq p$, on obtient $s' \in C_G \setminus \{p\} = C'$.

La proposition b) est vérifiée.

Montrons la contraposée de la proposition a). Si $C_G \notin \{\emptyset, \{p\}\}$, alors $C_G \setminus \{p\} \neq \emptyset$. Ainsi la proposition b) donne $C_{G'} = C_G \setminus \{p\} \neq \emptyset$.

La proposition a) est vérifiée.

Pour la proposition c) i), on suppose que $C_{G'}$ contient un élément c' et on montre la contraposée, c'est à dire :

$$C_G \notin \{\emptyset, \{p\}\} \Rightarrow (p, c') \in A$$

Si on suppose que $C_G \notin \{\emptyset, \{p\}\}$, alors $C_G \setminus \{p\} \neq \emptyset$ et la proposition b) indique que $C_{G'} = C_G \setminus \{p\}$. Comme $c' \in C_{G'} \subset C_G$ et que C_G est une clique de célébrités de G, on obtient $(p, c') \in A$.

La proposition c) i) est vérifiée.

Pour la proposition c) ii), on suppose que $(c', p) \notin A$. Dans un premier temps, on montre que $p \notin C_G$. En effet, si on suppose par l'absurde que $p \in C_G$, alors comme $(p, c') \in C_G \times S$ et que C_G est une clique de célébrités de G, on obtient $(c', p) \in A$ ce qui contredit l'hypothèse initiale. On a donc :

$$p \notin C_G$$

Pour conclure, il suffit de montrer que si $C_G \neq \emptyset$, alors $C_G = C_{G'}$. Si $C_G \neq \emptyset$, alors :

$$C_G \setminus \{p\} = C_G \neq \emptyset$$

et donc la proposition b) donne :

$$C_{G'} = C_G \setminus \{p\} = C_G$$

La proposition c) ii) est vérifiée.

Pour la proposition c) iii), supposons $(p, c') \in A$ et $(c', p) \in A$, et montrons que $C_G = \emptyset$ ou $C_G = \{p\} \cup C_{G'}$. On traite deux cas :

→ Le cas où $p \notin C_G$. Dans ce cas, on suppose par l'absurde que $C_G \neq \varnothing$. Alors $C_G \setminus \{p\} = C_G \neq \emptyset$ et la proposition b) donne $C_{G'} = C_G \setminus \{p\} = C_G$. Comme $(c', p) \in (C_{G'} \times S) = (C_G \times S)$ et que C_G est une clique de célébrités de G, on obtient :

$$(c', p) \in A \Rightarrow p \in C_G$$

Cette implication contredit nos hypothèses initiales.

→ Le cas où $p \in C_G$. Comme $(p, c') \in C_G \times S$ et que C_G est une clique de célébrités de G, on a :

$$(p, c') \in A \Rightarrow c' \in C_G.$$

Par hypothèse, on a $(p, c') \in A$ et donc $c' \in C_G$. Comme $c' \neq p$, on a $c' \in C_G \setminus \{p\}$ et donc $C_G \setminus \{p\} \neq \emptyset$. Ainsi, la proposition b) indique que :

$$C_{G'} = C_G \setminus \{p\}$$

Comme $p \in C_G$, on obtient $C_G = \{p\} \cup C_{G'}$.

La proposition c) iii) est vérifiée.

Question 13 –

```
# Cette fonction teste si e appartient à L.
def appartient(e, L):
    for x in L:
        if e == x:
            return True
    return False
```

```
def est_clique(L, R):
    for r1 in R:
        for r2 in R:
            if not appartient(r2, L[r1]):
                return False
    return True
```

Question 14 – On numérote les tours de boucle de $t = 1$ à $t = 7$ et on décrit chaque tour de boucle avec le tableau ci-dessous.

	Valeurs de s	Test ligne 6	Valeur de c	Test ligne 12	Test ligne 16	Test ligne 20	Valeur de C à la fin du tour
$t = 1$	$s = 0$	Vrai	Non définie	Non testé	Non testé	Non testé	[0]
$t = 2$	$s = 1$	Faux	$c = 0$	Faux	Faux	Vrai	[0, 1]
$t = 3$	$s = 2$	Faux	$c = 0$	Vrai	Vrai	Faux	[2]
$t = 4$	$s = 3$	Faux	$c = 2$	Faux	Vrai	Faux	[2]
$t = 5$	$s = 4$	Faux	$c = 2$	Faux	Faux	Vrai	[2,4]
$t = 6$	$s = 5$	Faux	$c = 2$	Faux	Vrai	Faux	[2,4]
$t = 7$	$s = 6$	Faux	$c = 2$	Faux	Faux	Vrai	[2,4,6]

Finalement la fonction renvoie la liste [2,4,6].

Question 15 –

```
def clique_possible_C(G):
    n = len(G)
    C = []
    S = [i for i in range(n)]
    for s in S:
        if len(C) == 0:
            C.append(s)
        else:
            c = C[0]
            t = False
            if not appartient(c, G[s]):
                C = [s]
                t = True
            if not appartient(s, G[c]):
                C = C
                t = True
            if not t:
                C.append(s)
    return C
```