SERIE de Revsion - C12-Développer en front end -

EXERCICE 1 :

On veut réaliser un composant « React » permettant de construire et d'afficher les trajets des trains. Un train a un « **id** » , un « **nom** » et une **liste des villes de passage** Chaque ville a un « **nom** » .

Chaque trajet des trains a un « **nomV** » et une « **ordreP** » enregistré dans le tableau **villes** :

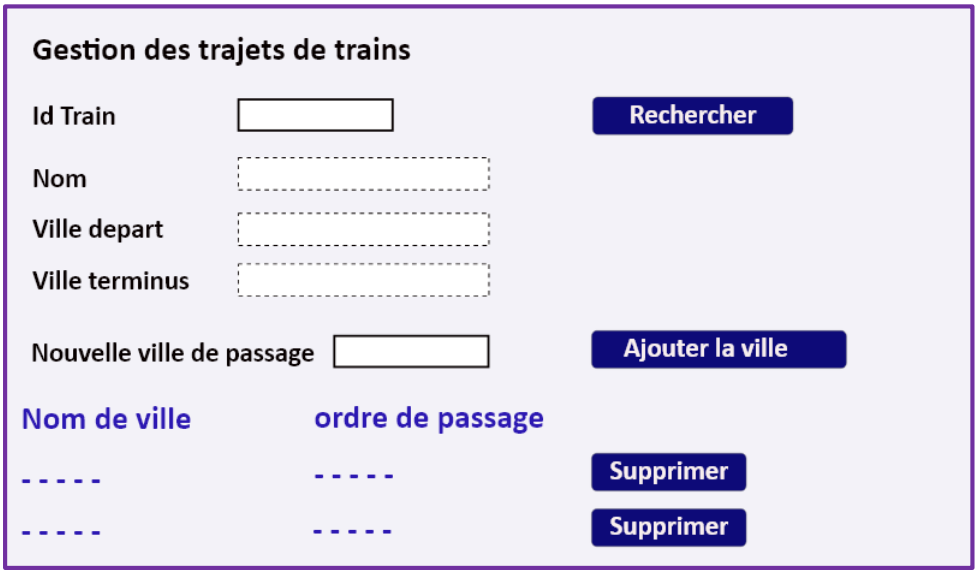
Exemple :

```
villes:[ {nomV :'fes', ordreP : 10}, {codeP:'meknes', ordreP : 20},{codeP:Kmisset, ordreP : 30}] .
```

Le composant utilise le tableau **trains** suivant :

```
const trains=[
    {id:'t1',nom:'Buraq',villes:[]},
    {id:'t2',nom:'Gazel', villes:[]},
    {id:'t3',nom:'Eagle-Wins', villes:[]},
    {id:'t4',nom:'Fennec', villes:[]},
]
```

On veut réaliser un Composant retournant le rendu suivant :



Questions :

1. En utilisant le tableau « **trains** » **4 Points / Question**
   a. Écrire le code du composant retournant le rendu ci-dessus.
   b. Déclarer les variables d'état nécessaires.
   c. Traiter le click du bouton « **Rechercher** »: la ville de départ est la première du tableau villes, la ville d'arrivée est la dernière.
   d. Le bouton « **Ajouter la ville** » permet d'ajouter la ville saisie au tableau « **villes** » du train en cours.
   e. Le bouton « **Supprimer** » supprime la ville du tableau « **villes** ».

EXERCICE2 :

On veut créer un composant « React » permettant de construire et d'afficher les trajets des lignes de bus. Une ligne de bus a un « **id** » , un « **nom** » et une **liste des arrêts** Chaque arrêt a un « **nomL** » et une « **ordreL** » enregistré dans le tableau **Lignes** :

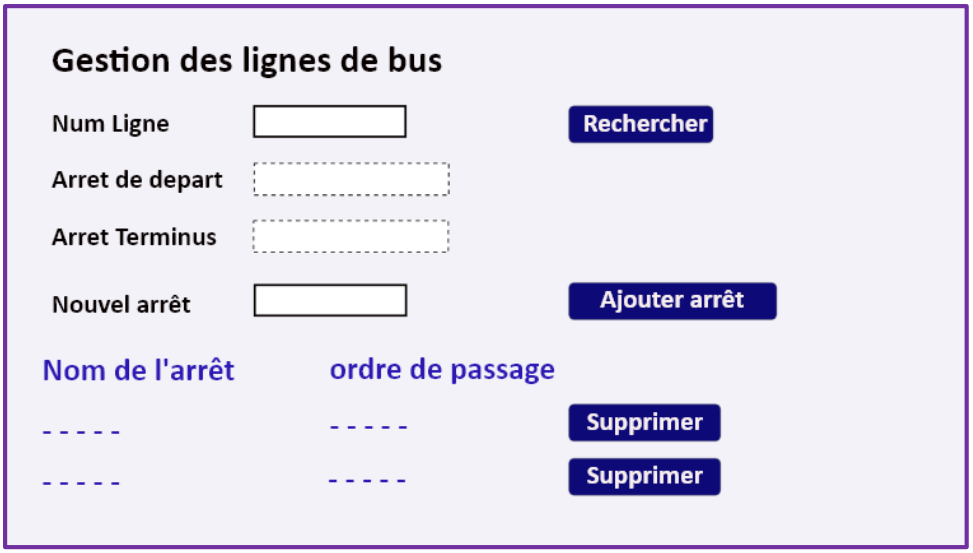
Exemple :

```
Arret :[{ codeA:'Adarissa', ordreA : 1}, {codeA: 'Saada', ordreA : 2},{codeA:
'Atlas', ordreA : 3}] .
```

Le composant utilise le tableau suivant :

```
const lignes=[
    {id:'l1', nom:Ligne1', arrets:[]},
    {id:'l2', nom:'Ligne2', arrets:[]},
    {id:'l3', nom:'Ligne3', arrets:[]},
    {id:'l4', nom:'Ligne4', arrets:[]},
]
```

On veut réaliser un Composant retournant le rendu suivant :



Questions :

En utilisant le tableau « **lignes** » **4 Points/Question**

a. Écrire le code du composant retournant le rendu ci-dessus.

b. Déclarer les variables d'état nécessaires.

c. Traiter le click du bouton « **Rechercher** » : L'Arrêt de départ est la première du **tableau Arrêt**, l'Arrêt du Terminus est la dernière du **tableau Arrêt**.

d. Le bouton **« Ajouter arrêt »** permet d'ajouter le nom de l'arrêt saisi dans le **tableau Arrêt** du bus en cours d'affichage.

e. Le bouton **« Supprimer »** supprime l'arrêt du **tableau Arrêts** du bus en cours.

EXERCICE 3 :

Une **Voiture électrique** est une voiture qui possède un « **code voiture** », une « **marque** », une « **autonomie** » exprimée en nombre de kilomètres et un « **kilométrage** » représentant le nombre de kilomètres parcourus. Pour chaque **Voiture électrique** on enregistre l'historique de ses parcours. Chaque parcours a un « **codeP** » et une « **distance** » enregistré dans le tableau `parcours_v` :

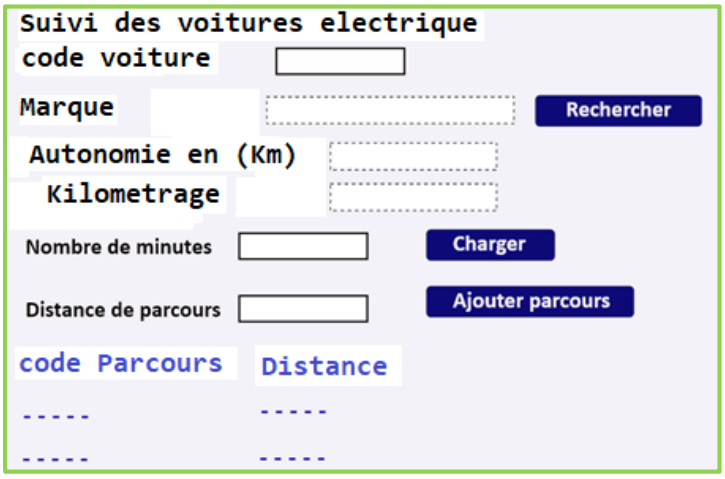
```
Exemple : parcours_v:[ {codeP:'1',distance: 20},{codeP:'2',distance: 30}] .
```

Pour manipuler ces voiture on utilise le tableau suivant :

```
const Voiture=[
    {codeVoiture:'v1',marque:'Voiture1',autonomie:0,kilometrage:0, parcours_v:[]},
    {codeVoiture:'v2',marque: 'Voiture2',autonomie:0,kilometrage:0, parcours_v:[]},
    {codeVoiture:'v3',marque: 'Voiture3',autonomie:0,kilometrage:0, parcours_v:[]},
    {codeVoiture:'v4',marque: 'Voiture4',autonomie:0,kilometrage:0, parcours_v:[]},
]
```

(Au départ le tableau `parcours_v` est vide car il y a pas de parcours)

On veut réaliser un Composant retournant le rendu suivant :



Chargement du vélo :

On suppose que chaque heure donne 50 Km d'autonomie. Et que l'autonomie maximale est de 500Km

Questions :

1. En utilisant le tableau « Voiture »
    a. Écrire le code du composant retournant le rendu ci-dessus.
    b. Déclarer les variables d'état nécessaires.
    c. Traiter le click du bouton « **Rechercher** » qui cherche les voitures par le « **code Voiture** »
    d. Le bouton « Charger » permet de convertir le nombre de minutes saisies en kilomètres et de l'ajouter au « **kilometrage** » de la voiture en cours d'affichage.
    e. Le bouton « **Ajouter parcours** » crée un nouveau parcours et l'ajoute au tableau parcours du Voiture en cours d'affichage (vérifier que l'autonomie est suffisante), lorsqu'on fait un parcours la distance est retranchée du « **kilometrage**

EXERCICE 4 :

Dans une application de compétition d'athlétisme nous avons un ensemble de courses dans lesquelles participent des conquérants.

Les courses sont enregistrées dans le tableau suivant :

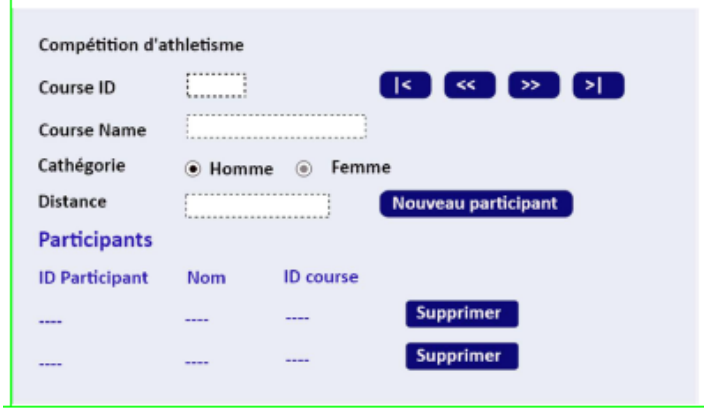
```
const courses=[
    {id: "MarcheF",nom:"Marche femmes",categorie:"F", distance:10000
    },
    {id: "MarcheH",nom:"Marche hommes",categorie:"H", distance:10000
    },
    {id: "MarathonF",nom:"Marathon fommes",categorie:"F", distance:42000
    },
    {id: "MarathonH",nom:"Marathon hommes",categorie:"H", distance:42000
    }
]
```

Les participants sont enregistrés dans un autre tableau participants.
Chaque participant a un id, un nom, idCourse.
Exemple :

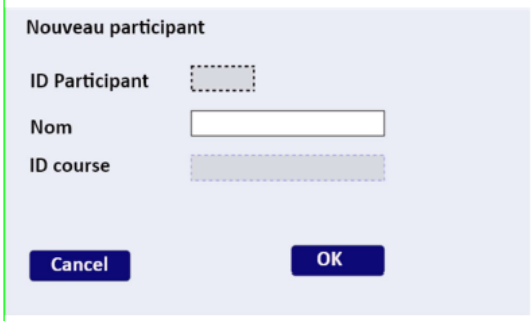
```
const participants=[
    {id: "1", nom:"Med",  idCourse: "MarcheH"},
    {id: "2",nom:"Rajae", idCourse: "MarcheF"}
    ]
```

L'id du participant est calculé automatiquement et égale à max id +1

Creer un composant React « Course » retournant le rendu suivant :



- Les boutons de navigation permettent la navigation entre les courses.
- Pour chaque course on affiche ses informations ainsi que la liste de ses participants.
- Le bouton supprimer permet de supprimer le participant correspondant du tableau participants.
- Sur « Click » du bouton « Nouveau participant » l'écran suivant est affiché



- Cet écran permet d'ajouter un participant à la course.
- L'id de participant est calculé automatiquement (max id +1).
- Le nom du participant est saisi dans la zone de texte.
- L'id de course est affiché automatiquement.

EXERCICE 5 :

On veut réaliser un composant « React » simulant un distributeur de boissons.

Chaque distributeur a un **id**, une **marque**, le **nombre de verres vides** et peut servir trois **boissons**.

Chaque boisson a un **nom**, un **prix** et une **quantité**.

L'application utilise le tableau suivant :

```
const t=[
  {id:1,marque:"Bosh",nombreVerre:10, boissons:[{nom:"Café",prix:10,qte:50},
        {nom:"Thé",prix:8,qte:20},  {nom:"Chocolat",prix:12,qte:40}  ]},
  {id:2,marque:"Samsung",nombreVerre:20, boissons:[{nom:"Café",prix:10,qte:4},
        {nom:"Thé",prix:8,qte:20},  {nom:"Chocolat",prix:12,qte:30}  ]},
  {id:3,marque:"Toshiba",nombreVerre:5, boissons:[{nom:"Café",prix:10,qte:2},
        {nom:"Thé",prix:8,qte:10},  {nom:"Chocolat",prix:12,qte:5}  ]}
]
```

En utilisant le tableau **t** réaliser un composant « React » qui retourne le rendu suivant :

Question :

-Utiliser un type de composant de votre choix (c'est-à-dire « class » ou « function »} et réaliser le rendu JSX. Il n'est pas demandé de donner un style au rendu de votre composant.

-Proposer une solution pour gérer l'état de votre composant.

-Le « click » sur le bouton « Rechercher » effectue une recherche dans le tableau **t** et affiche les informations du distributeur et la liste de ses boissons.

-Si la quantité de boisson ou le nombre de verres sont égaux à 0 le bouton « Servez-vous » correspondant est désactivé.

-Le « click » sur le bouton « Servez-vous » décrémente de 1 la quantité de la boisson correspondante ainsi que le nombre de verre du distributeur en cours.

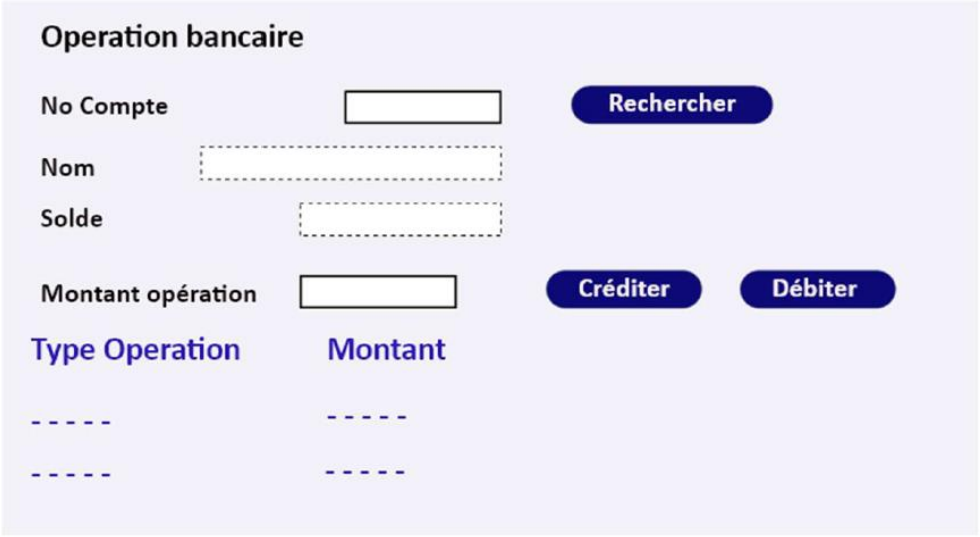
EXERCICE 6 :

On veut réaliser un composant « React » pour enregistrer des opérations sur des comptes bancaires. Chaque compte a un **numéro**, un **nom de client**, un **solde** et une **liste des opérations**. Une opération bancaire est caractérisée par un **type d'opération** et un **montant**.
Le type d'opération est soit « **Crédit** » ou « **Débit** ». Crédit veut dire que le montant de l'opération est ajouté au solde, « Débit » le montant de l'opération est retiré du solde.

Le composant « React » utilise le tableau suivant :

```
const comptes=[
    {num:1,nom:"Ali",solde:18000, operations:[
        {type:"Credit",montant:10000},  {type:"Credit",montant:8000},  ]},
    {num:1,nom:"Sarah",solde:4000, operations:[
        {type:"Credit",montant:6000},  {type:"Debit",montant:2000},  ]}
]
```

En utilisant le tableau **comptes** réaliser un composant « React » qui retourne le rendu suivant :



Question :

-Utiliser un type de composant de votre choix (c'est-à-dire « class » ou « function »} et réaliser le rendu JSX. Il n'est pas demandé de donner un style au rendu de votre composant.

-Proposer une solution pour gérer l'état de votre composant.

-Le « click » sur le bouton « Rechercher » effectue une recherche dans le tableau **comptes** et affiche les informations du compte et la liste de ses opérations

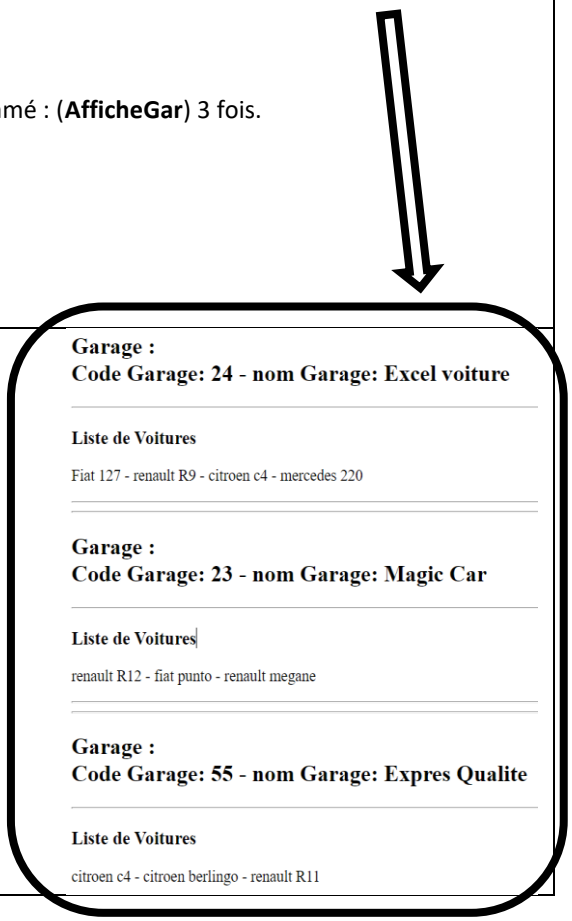
-Le bouton « Créditer » : enregistre une opération de crédit ayant le montant saisi, l'ajoute à la liste des opérations du compte et actualise le solde.

-Le bouton « Débiter » : vérifie la disponibilité du solde, si le solde est suffisant, une opération de débit est ajoutée à la liste des opérations du compte et le solde est actualisé.

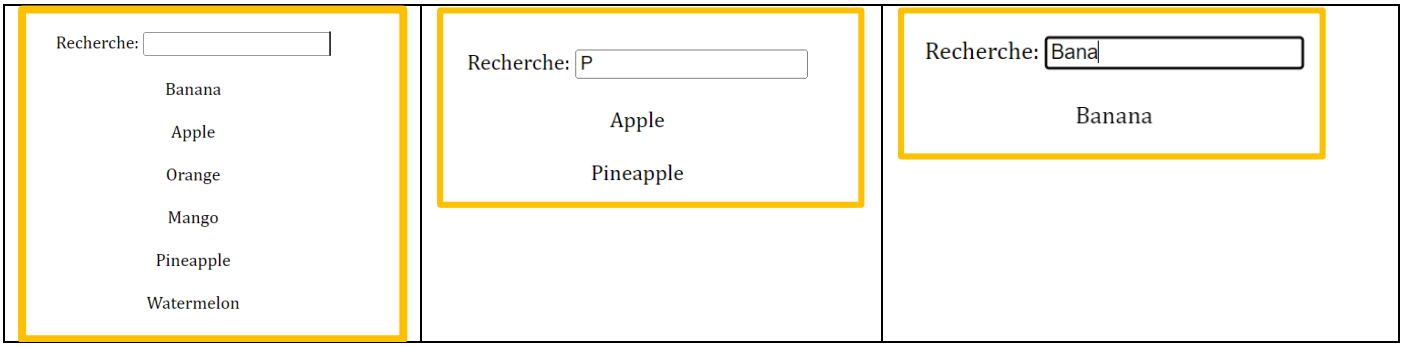
EXERCICE 7 :

On souhaite afficher le même composant plusieurs fois mais avec des informations différentes, en Utilisant l'Object **props** :
props.Garage.CodeG , props.Garage.NomG , props.ListeVoitures
ListeVoitures est un tableau de voitures ex :
[" Citroen c4 ", " Citroen berlingo ", " Renault R11 "]
Ce composant **AfficheGar** va nous permettre de passer les informations nécessaires au composant qui va afficher les informations suivantes :

a) Creer ce composant nommé : (**AfficheGar)**
b) Creer le composant nommé : (**App**) qui va consommer le composant nommé : (**AfficheGar**) 3 fois.



EXERCICE 8 :



Soit le tableau suivant :

```
const list = [
   "Banana",
   "Apple",
   "Orange",
   "Mango",
   "Pineapple",
   "Watermelon"
 ];
```

Ecrire un composant fonctionnel de filtre et de recherche simple pour afficher une liste filtrée en fonction de la requête de recherche saisie par l'utilisateur.

Voici les étapes pour créer un filtre de recherche à l'aide de React JS :

1) Déclarez les états de réaction pour les valeurs d'entrée de recherche.
2) Créez un texte d'entrée HTML pour entrer le terme de recherche et mettre à jour l'état dans la fonction **onChange** .
3) Ajoutez **Array.filter()** sur la liste des éléments avec la valeur du terme de recherche.

EXERCICE 9 :

Contrôle continu

Exercice I :

Une API nous envoie un tableau d' objet JSON (Buteurs) de la forme :

```
Buteurs = [ {   "id": 1,
                "player": {"player_id": 19779, "player_name": "Karim Benzema"},
                "team": "Real Madrid" ,
                "goals": {"home": 7,  "away": 7 },
                },
                { "id": 2,
                  "player": {"player_id": 4934, "player_name": "Robert Lewandowski" },
                   "team": "FC Bayern Munich",
                    "goals": {"home": 9,  "away": 4},

                },
                { . . . }
                ] ;
```

1) Ecrire le code JS qui renvoie le nombre de buts marqués par le joueur qui a le nom Mohamed Salah
2) Ecrire le code JS qui renvoie les noms des joueurs qui ont marqué plus que 3 buts à l'extérieur (away)
3) Ecrire le code JS qui renvoie le plus grand nombre de buts marqués à l'intérieur (home)
4) Ecrire le code JS qui renvoie la somme des buts qui sont marqués à l'extérieur (away)
5) Ecrire le code JS qui renvoie le nom de l'équipe qui a marqué le plus grand nombre de buts à l'extérieur (away)
6) Ecrire le code JS qui créée un autre tableau Buteurs2 contenant les éléments du tableau Buteurs en ajoutant l'élément suivant à la fin :

```
{ "id": 21,
  "player": {"player_id": 500, "player_name": "Halland" },
  "team": "M. City" ,
  "goals": {"home": 9,  "away": 9}
}
```

Exercice II :

On veut réaliser un composant reactJs appelé *Compteur*. Ce composant contient un élément h2 qui affiche la valeur du compteur et un bouton « *Incrementer* » qui permet, après clic, d'augmenter la valeur du compteur de 2.

7) Ecrire le code du composant *Compteur* de type classe
8) Réécrire le composant *Compteur*, mais de type fonction avec en plus :

- La valeur de l'incrémentation du compteur est aléatoire (à la place de 2)
- La valeur du compteur est affichée avec une couleur qui est passée en props à ce composant
- Quand la valeur du compteur devient égale à 50, la couleur devient rouge

EXERCICE 10 :

Contrôle continu

Exercice I :

On considère l'objet suivant :

```
produits = [ {id :11, nom : "clé USB" , prix : 70 , famille:  "INFO"},
             {id :22, nom : "Table" , prix : 250 , famille:  "Mobilier"},
             {id :33, nom : "Souris" , prix : 40 , famille:  "INFO"},
             {id :44, nom : "Chaise" , prix : 170 , famille:  "Mobilier"},
             {id :55, nom : "Pince" , prix : 35 , famille:  "INFO"},
             {id :66, nom : "Routeur" , prix : 300 , famille:  "INFO"},,

             .....

             ]
```

1) Ecrire le code JS qui retourne les produits de la famille INFO qui ont le prix supérieure à 80
2) Ecrire le code JS qui calcule la moyenne des prix des produits
3) Ecrire le code JS qui détermine le nom du produit le plus cher
4) Ecrire le code JS qui créée un autre tableau *produits2* contenant les éléments du tableau *produits* en supprimant le produit qui a le ID égal à 33

Exercice II :

5) Créer un composant de type classe appelé ***Inscription*** qui contient une zone de texte pour saisir le mot de passe et une autre zone de texte pour confirmer le mot de passe et un bouton « Valider ».

Au clic sur le bouton Valider, on doit vérifier la saisie :

- Si les mots de passe saisis ne sont pas identiques, on affiche en rouge le message "Mot de passe saisi ne coïncide pas avec le mot de passe confirmé "
- Si les 2 mots de passe saisis sont identiques, on affiche en vert le message "Bienvenu"

Mot de passe : 123 Confirmer : 456 Valider

Mot de passe saisi ne coïncide pas avec le mot de passe confirmé

6) Réécrire le même composant ***Inscription*** , mais de type fonction avec en plus la vérification si le mot de passe est « faible », « moyen » ou « fort »:

Après clic sur le bouton « Valider » , si les 2 mots de passe saisis sont identiques, on affiche le message :

- "Bienvenu, mot de passe faible " si la taille du mot de passe est inférieure à 8
- "Bienvenu, mot de passe moyen " si la taille du mot de passe est supérieure à 8
- "Bienvenu, mot de passe fort " si la taille du mot de passe est supérieure à 8 et contient un mélange de lettres majuscules et minuscules.

EXERCICE 11 :

OFPPT

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle
et de la Promotion du Travail

Direction Régionale de Casa Settat

EVALUATION DE FIN DE MODULE REGIONAL
AU TITRE DE L'ANNEE : 2022/2023

Filière : DDOWFS	Année de formation : 2A
Niveau : TS	Epreuve : TH
N° du module : M204	Durée : 3H
Intitulé du module : développement front end	Coefficient :3
Date d'évaluation : 20/12/202	Barème/40

Variante : V1

Le but de cet exercice est de vous faire réaliser une application composée :

1. d'un store redux permettant de gérer l'état global de l'application : un ensemble de pays.

✓ le store a un état initial contenant un tableau avec les données sur les continents .Pour chaque continent, on doit disposer des informations suivantes :

- Code: code du continent,
- Name: nom du continent,
- SurfaceArea: superficie,
- avatar :url
- Population: population totale.
- pays: tableau décrivant les pays du continent
  - Name: nom du pays,
  - Population: population du pays.
  - Capital : la capitale du pays
  - IndepYear: année d'indépendance (si absent : pays dépendant d'un autre).
  - Image :url

et de(s)reducer(s) permettant l'accès aux données, l'ajout d'un pays ou la modification de la population d'un continent

✓ des composants react qui accèdent au store pour afficher les données, les ajouter, les modifier ou les filtrer.

Travail à faire

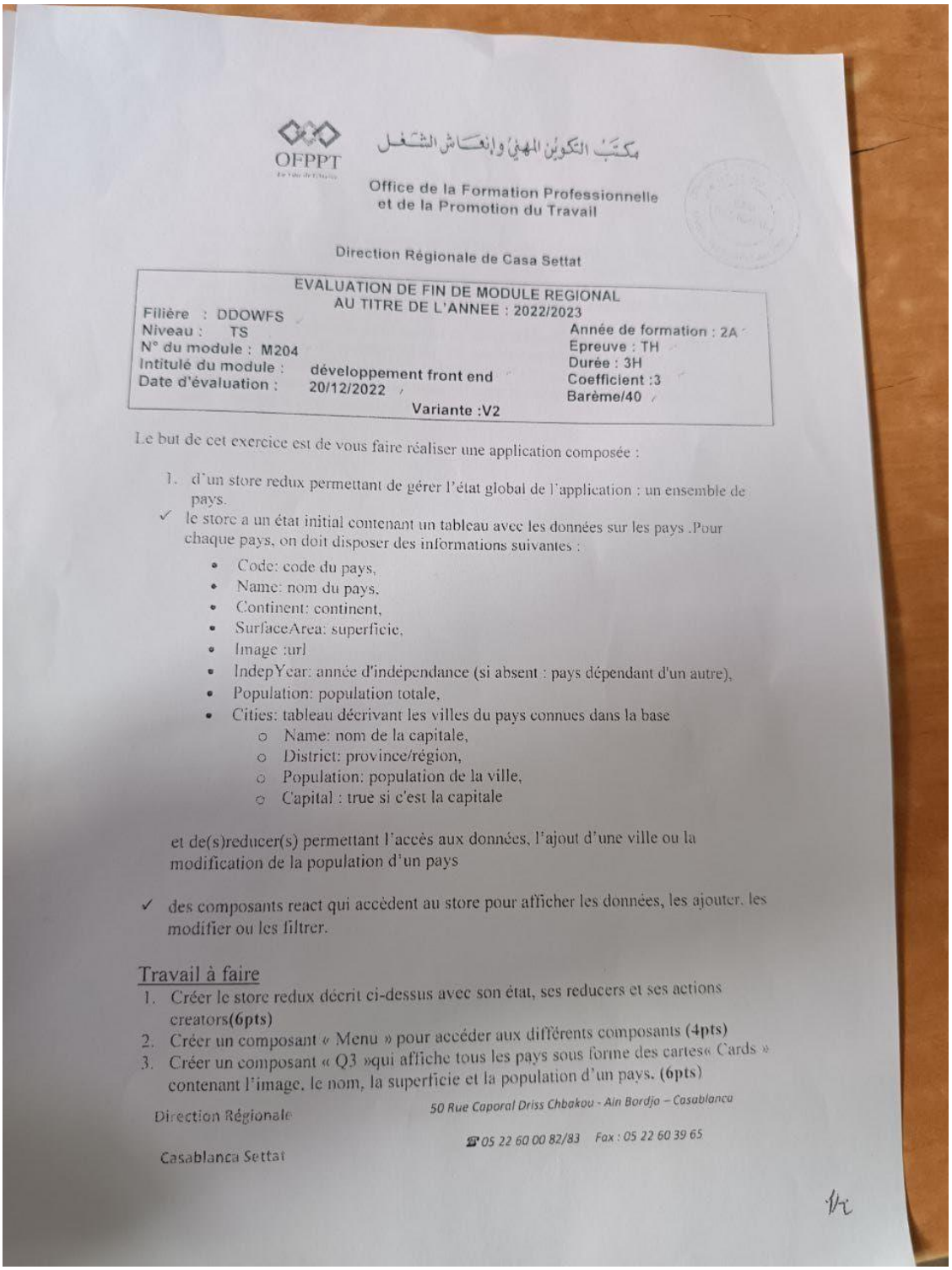
1. Créer le store redux décrit ci-dessus avec son état, ses reducers et ses actions creators(6pts)
2. Créer un composant « Menu » pour accéder aux différents composants (4pts)
3. Créer un composant « Q3 »qui affiche tous les continents sous forme des cartes (CARD) contenant l'image, le nom, la superficie et la population d'un continent. (6pts)

Direction Régionale
Casablanca Settat

50 Rue Caporal Driss Chbakou - Ain Borja - Casablanca
☎ 05 22 60 00 82/83 Fax : 05 22 60 39 65

4. Créer un composant « Q4 » qui affiche les années de l'indépendance sous forme des liens hypertextes.au clic sur un lien, on affiche dans le composant Q3 les pays qui ont obtenu l'indépendance dans l'année choisie **(6pts)**
5. Créer un composant « Q5 » permet d'ajouter une ville ou de modifier sa population. **(6pts)**
6. Créer un composant « Q6 » qui permet de filtrer les pays par population ou par continent (le choix du continent se fait via une liste déroutante) **(6pts)**
7. Soit "https://FAKEAPI.com/PAYS"» l'URL de l'API qui retourne la liste de tous les pays, ajouter dans votre store une fonction asynchrone qui rempli le tableau pays de votre state par les pays fournis par l' API **(6pts)**

EXERCICE 12 :



OFPPT

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle
et de la Promotion du Travail

Direction Régionale de Casa Settat

EVALUATION DE FIN DE MODULE REGIONAL
AU TITRE DE L'ANNEE : 2022/2023

Filière : DDOWFS			Année de formation : 2A
Niveau : TS			Epreuve : TH
N° du module : M204			Durée : 3H
Intitulé du module :	développement front end		Coefficient :3
Date d'évaluation :	20/12/2022		Barème/40

Variante :V2

Le but de cet exercice est de vous faire réaliser une application composée :

1. d'un store redux permettant de gérer l'état global de l'application : un ensemble de pays.

✓ le store a un état initial contenant un tableau avec les données sur les pays .Pour chaque pays, on doit disposer des informations suivantes :

- Code: code du pays,
- Name: nom du pays,
- Continent: continent,
- SurfaceArea: superficie,
- Image :url
- IndepYear: année d'indépendance (si absent : pays dépendant d'un autre),
- Population: population totale,
- Cities: tableau décrivant les villes du pays connues dans la base
  - Name: nom de la capitale,
  - District: province/région,
  - Population: population de la ville,
  - Capital : true si c'est la capitale

et de(s)reducer(s) permettant l'accès aux données, l'ajout d'une ville ou la modification de la population d'un pays

✓ des composants react qui accèdent au store pour afficher les données, les ajouter, les modifier ou les filtrer.

Travail à faire

1. Créer le store redux décrit ci-dessus avec son état, ses reducers et ses actions creators**(6pts)**
2. Créer un composant « Menu » pour accéder aux différents composants **(4pts)**
3. Créer un composant « Q3 »qui affiche tous les pays sous forme des cartes« Cards » contenant l'image, le nom, la superficie et la population d'un pays. **(6pts)**

Direction Régionale
Casablanca Settat
50 Rue Caporal Driss Chbakou - Ain Bordjo – Casablanca
05 22 60 00 82/83 Fax : 05 22 60 39 65

4. Créer un composant « Q4 » qui affiche les années de l'indépendance sous forme des liens hypertextes.au clic sur un lien, on affiche dans un autre composant les pays qui ont obtenu l'indépendance dans l'année cliquée **(6pts)**
5. Créer un composant « Q5 » permet d'ajouter un pays ou de modifier sa population. **(6pts)**
6. Créer un composant « Q6 » qui permet de filtrer les pays par population ou par continent (le choix du continent se fait via liste déroulante) **(6pts)**
7. Soit "https://FAKEAPI.com/continents"» l'URL de l'API qui retourne la liste des continents, ajouter dans votre store une fonction asynchrone qui rempli le tableau continent de votre state par les continents fournis par l'API **(6pts)**