DS n° 1 Informatique III

Calculatrice et documents non autorisés

Lorsqu'aucune consigne n'est précisée, les réponses peuvent être données en langage C ou en pseudo-code.
Vous pouvez définir autant de fonctions que vous le souhaitez dans chaque exercice. Il n'y a pas nécessairement l'équivalence : une question = une fonction.

Attention : Votre code doit être robuste. Vous n'êtes cependant pas obligés de vérifier l'integralité des données des structures vues en CM (définir et utiliser la fonction `verifierPile()` par exemple n'est pas obligatoire). 2 pt bonus vous seront cependant accordés si vous faites correctement la vérification d'intégrité des données des structures.

Exercice 1 (*File d'attente dans un parc d'attractions*) (11 pts)

Au parc Vercingetorix, l'attraction Ose-isis accueille de nombreux visiteurs. Dans cet exercice, nous allons simuler sa file d'attente qui donc être gérée comme une file dynamique. Les visiteurs viennent par groupe et embarquent sur l'attraction en même temps que leurs amis. La file stockera des entiers correspondant au nombre de personnes appartenant à un groupe voulant prendre place dans l'attraction.

1. Déclarer la/les structures `File` permettant de gerer une telle file.
2. Les visiteurs peuvent appartenir à des groupes entre 2 et 10 personnes inclues. Ce nombre sera tiré aléatoirement. Écrire une fonction/procédure `entrerFile` qui simule l'entrée d'un nouveau groupe de personnes dans la file d'attente de l'attraction.

   Écrire une fonction/procédure `File* sortirFile(File* pFile, int* groupe)` qui va défiler un groupe de personnes et qui va retourner la nouvelle file ET la taille du groupe qui est sorti de la file.
3. Le bâtiment de la file d'attente ne peut accueillir que 500 personnes au maximum. Écrire une fonction/procédure `filePleine(...)` qui renvera 1 si la file d'attente est pleine, 0 sinon.

   Les trains de l'attraction Ose-isis sont constitué de 8 rangées de 4 sièges chacun. Les règles de placement sont les suivantes :
   — Si un groupe fait 4 personnes ou moins, il prendra une rangée
   — S'il fait entre 5 et 8 personnes inclues, il prendra deux rangées.
   — Si le groupe fait plus de 8 personnes il prendra trois rangées.
   — Les places vides dans les rangées ne seront pas comblées.
   — Les groupes ne peuvent pas être séparés : s'il reste une rangée vide dans le train mais que le prochain groupe dans la file fait plus que 4 personnes, le train partira avec une rangée vide.
4. Écrire la fonction/procédure `remplirTrain` qui va vider la file d'attente jusqu'à avoir rempli au mieux toutes les rangées du train. La fonction devra afficher également le nombre de places vides du train après ce remplissage.
5. En 3 minutes, en moyenne, 2 trains sont remplis et font le parcours de l'attraction, et 16 groupes de personnes rentrent dans la file d'attente (s'il reste de la place dans le bâtiment). A l'aide des fonctions précédentes, écrire la fonction/procédure `three_minutes` qui va simuler l'avancement de la file sur une durée de 3 minutes.

Exercice 2 (*Structure DEQUE*) **(9 pts)**
Il existe une structure appelée "deque" (pour l'anglais "Double-Ended-QUEue" : "File à double extrémité"). Cette structure s'apparente à une **liste dynamique doublement chaînée** dont on connait à chaque instant :

— le premier chainon
— le dernier chainon
— sa taille (son nombre d'éléments)

Cette structure permet d'effectuer des ajouts et des suppressions à chaque extrémité de la file avec une complexité temporelle optimisée : elle réunit donc les avantages des Files et des Piles.

Il existe plusieurs usages pour ce type de structure : nous allons la choisir pour gérer un historique des adresses des sites Internet visités dans un navigateur.

1. Définir la structure d'un chainon avec comme élément une chaine de caractères allouée dynamiquement : cette chaine contiendrait l'URL d'un site Internet dans notre exemple.
2. Définir la structure "Deque" pour qu'elle réponde aux critères décrits précédemment.
3. On prend l'hypothèse que les fonctions de création et de suppression de chainon existent :**vous n'avez donc pas besoin de les coder**.
   — `creerChainon` : elle prend en paramètre une chaine de caractères, puis retourne le chainon créé.
   — `supprimerChainon` : elle prend en paramètre un chainon qu'elle libère et ne retourne rien.
   Créer la fonction/procédure `ajoutDebutDeque` qui prend en paramètre une structure `Deque` ainsi qu'une chaine de caractères et qui va ajouter un chainon en début.
4. Créer la fonction/procédure `supprimerFinDeque` qui prend en paramètre une structure Deque et qui va simplement supprimer le dernier élément
5. On imagine que l'historique du navigateur Internet que nous utilisons ne peut contenir que 10 adresses de sites. Créer la fonction/procédure `ajoutHistorique` qui va ajouter une adresse passée en paramètre en début de liste, et va regarder si la taille de la liste depasse 10 : si c'est le cas, le dernier chainon devra être supprimé.
6. Écrire une fonction `precedent` qui va prendre en paramètre une structure `Deque` et qui retournera l'URL du site Internet que le navigateur doit ouvrir si l'utilisateur clique sur "précédent" (retourner sur la page précédente). On ne demande pas de modifier l'historique ici.