Solutions to C Programming Exercises

Partie 1 : Questions à Choix Multiples (QCM)

1. **Qu'est-ce qu'un pointeur en C?**

   - **a) Une variable qui stocke l'adresse d'une autre variable.**
     Explication : Un pointeur est une variable qui contient l'adresse mémoire d'une autre variable.

2. **Quel est le résultat de la fonction récursive suivante?**

```
int factorial(int n) {
    if (n == 0) return 1;
    else return n * factorial(n - 1);
}
int main() {
    printf("%d", factorial(5));
    return 0;
}
```

   - **a) 120**
     Explication : La fonction calcule la factorielle de 5, qui est $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

3. **Dans une liste simplement chaînée, à quoi pointe le pointeur `next` du dernier nœud?**

   - **c) NULL**
     Explication : Dans une liste simplement chaînée, le dernier nœud pointe vers `NULL` pour indiquer la fin de la liste.

Partie 2 : Questions Vrai ou Faux

1. **Une structure en C peut contenir des membres de types de données différents.**
   Vrai
   Explication : Une structure en C peut contenir des membres de différents

types de données, comme des entiers, des flottants, des chaînes de caractères, etc.

2. **Dans une pile (stack), le dernier élément ajouté est le premier à être retiré (LIFO).**
   Vrai
   Explication : Une pile fonctionne sur le principe LIFO (Last In, First Out), ce qui signifie que le dernier élément ajouté est le premier à être retiré.

3. **Une fonction récursive doit toujours avoir un cas de base pour éviter une récursion infinie.**
   Vrai
   Explication : Un cas de base est nécessaire pour arrêter la récursion et éviter qu'elle ne continue indéfiniment.

Partie 3 : Exercices de Programmation

Exercice sur les Pointeurs et les Structures

Écrivez un programme en C qui définit une structure `Student` avec les membres suivants :

- `name` (chaîne de caractères)
- `age` (entier)
- `grade` (flottant)

Créez une fonction `displayStudent` qui prend un pointeur vers une structure `Student` et affiche ses détails.

```
#include <stdio.h>
#include <string.h>

// Définition de la structure Student
struct Student {
    char name[50];
    int age;
    float grade;
};

// Fonction pour afficher les détails d'un étudiant
void displayStudent(struct Student *student) {
    printf("Name: %s\n", student->name);
    printf("Age: %d\n", student->age);
    printf("Grade: %.2f\n", student->grade);
}
```

```
int main() {
    // Création d'une instance de la structure Student
    struct Student student1;
    strcpy(student1.name, "John Doe");
    student1.age = 20;
    student1.grade = 88.5;

    // Affichage des détails de l'étudiant
    displayStudent(&student1);

    return 0;
}
```

Exercice sur les Fonctions Récursives

Écrivez une fonction récursive en C pour calculer $n! \times m!$.

```
#include <stdio.h>

// Fonction récursive pour calculer la factorielle
int factorial(int n) {
    if (n == 0) return 1;
    else return n * factorial(n - 1);
}

// Fonction pour calculer n! × m!
int factorialProduct(int n, int m) {
    return factorial(n) * factorial(m);
}

int main() {
    int n = 5, m = 3;
    printf("Factorial product of %d and %d is %d\n", n, m, factorialProduct(n, m));
    return 0;
}
```

Exercice sur les Listes Chaînées

Écrivez un programme en C pour créer une liste simplement chaînée avec 3 nœuds. Chaque nœud doit stocker une valeur entière. Affichez les valeurs de tous les nœuds de la liste.

```
#include <stdio.h>
#include <stdlib.h>
```

```
// Définition de la structure Node
struct Node {
    int data;
    struct Node* next;
};

// Fonction pour afficher les valeurs de la liste chaînée
void printList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    // Création de 3 nœuds
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;

    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    // Assignation des valeurs et liaison des nœuds
    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = NULL;

    // Affichage de la liste chaînée
    printList(head);

    // Libération de la mémoire
    free(head);
    free(second);
    free(third);

    return 0;
}
```