Partie I : Théorique (4 pts)

1. Citer 6 fonctions de manipulation des chaines de caractères de la bibliothèque **string.h**
   1. **strlen**
   2. **strcpy**
   3. **strcmp**
   4. **strcat**
   5. **strstr**
   6. **strchr**
2. Lors de l'appel d'une fonction le tableau en paramètre représente**un pointeur qui pointe sur le premier élément du tableau.**

Partie II : Pratique (38 pts)

Exercice 1 : (8 pts)

#include<stdio.h>

//Question a

**int** suite(**int** n){

if(n == 1 || n == 2)

return n;

return (suite(n-1) \* suite(n-2));

}

//Question b

void suite\_produit(**int** n){

**int** i, s, p = 1;

if(n < 1){

printf("Suite non definie pour le terme : %d \n", n);

}

else{

for(i=1; i<=n; i++){

s = suite(i);

p = p \* s;

printf("U(%i) = %d\n", i, s);

}

printf("Le produit des termes : %d \n", p);

}

}

//Question c

main(){

**int** n, nb, i, t, s = 0;

//Appel de la fonction a

printf("Entrer un entier: ");

scanf("%d", &n);

s = suite(n);

printf("U(%d) = : %d\n",n, s);

//Appel de la fonction b

printf("Entrer le nombre de termes: ");

scanf("%d", &nb);

suite\_produit(nb);

}

**Exemple d’exécution :**

Entrer un entier: 6  
U(6) = : 32  
Entrer le nombre de termes: 5  
U(1) = 1  
U(2) = 2  
U(3) = 2  
U(4) = 4  
U(5) = 8  
Le produit des termes : 128

Exercice 2 : (28 pts)

#include<stdio.h>

#include<string.h>

//Question 1 : Définition de la structure Salle

typedef struct {

**int** id;

char libelle[20];

**int** capacite;

} **Salle**;

//Question 2 : Fonction qui cree une salle

//Cette fonction est améliorer dans la version 2

**Salle** **CreerSalle**(**int** idSalle,char \*libelleSalle,**int** capacite) {

**Salle** s;

s.id = idSalle;

strcpy(s.libelle, libelleSalle);

s.capacite = capacite;

return s;

}

//Question 3 : Fonction qui remplit un tableau de salles

//Cette fonction est améliorer dans la version 2

void **SaisirInfosSalle**(**Salle** **Liste**[],**int** n) {

**int** i, id, capacite;

char libelle[20];

for(i=0; i<n; i++){

printf("Salle : %d : \n", i+1);

printf("Entrer l' identificateur : ");

scanf("%d", &id);

printf("Entrer le libelle : ");

scanf("%s", libelle);

printf("Entrer la capacite : ");

scanf("%d", &capacite);

**Liste**[i] = **CreerSalle**(id, libelle, capacite);

}

}

//Question 2 - Version 2: Fonction qui cree une salle

**Salle** **CreerSalle2**() {

**Salle** s;

printf("Entrer l'' identificateur : ");

scanf("%d", &s.id);

printf("Entrer le libelle : ");

scanf("%s", s.libelle);

printf("Entrer la capacite : ");

scanf("%d", &s.capacite);

return s;

}

//Question 3 - Version 2: Fonction qui remplit un tableau de salles

void **SaisirInfosSalle2**(**Salle** **Liste**[],**int** n) {

**int** i;

for(i=0; i<n; i++){

**Liste**[i] = **CreerSalle2**();

}

}

//Question 4 : Affichage d'une salle

void **AfficherSalle**(**Salle** s) {

printf("Id Salle : %d; ", s.id);

printf("Libelle Salle : %s; ", s.libelle);

printf("Capacite Salle : %d\n\n", s.capacite);

}

//Question 5 : Affichage de la liste des salles

void **AfficherInfosSalles**(**Salle** **Liste**[], **int** n ) {

**int** i;

for(i=0; i<n; i++){

printf("Salle %d : \n", i+1);

**AfficherSalle**(**Liste**[i]);

}

}

//Question 6 : Recherche d'une salle par ID

//Le type bool n'existe pas en langage C (il est introduit en langage C++)

**int** **RechercherSalle**(**int** id, **Salle** **Liste**[], **int** n){

**int** v = 0, i;

for(i=0; i<n; i++){

if(**Liste**[i].id == id){

v = 1;

break;

}

}

return v;

}

//Question 7 : Suppression d'un salle

void **SupprimerSalle**(**int** id, **Salle** **Liste**[], **int** \*n){

**int** i, j;

if(!**RechercherSalle**(id, **Liste**, \*n)){

printf("Salle n''existe pas!\n");

}

else{

//On suppose que l'id est unique

for(i=0; i<\*n; i++){

if(**Liste**[i].id == id){

for(j=i; j<(\*n)-1; j++){

**Liste**[j] = **Liste**[j+1];

}

(\*n)--;

//i--; si l'id n'est pas unique

}

}

printf("Liste des salles apres suppression de la salle %d : \n", id);

**AfficherInfosSalles**(**Liste**, \*n);

}

}

//Question 8 : Tri de la liste des salles par ordre (Croissant) alphabétique des libelles

void **TrierSallesParLibelle**(**Salle** **Liste**[], **int** n){

**int** i, j;

**Salle** s;

for(i=0; i<n; i++){

for(j=i;j<n;j++){

if(strcmp(**Liste**[i].libelle , **Liste**[j].libelle) == 1)

{

s = **Liste**[i];

**Liste**[i] = **Liste**[j];

**Liste**[j] = s;

}

}

}

puts("Liste des Salles triees par libelle:");

**AfficherInfosSalles**(**Liste**, n);

}

//Question 9 : Sauvegrade de la liste des salles dans un fichier texte

void **Sauvegarder** (**Salle** **Liste**[], **int** n){

**FILE** \*f;

**int** i;

f = fopen("Salles.txt", "w");

if(!f){

printf("Erreur de creation du fichier!");

return;

}

for(i=0; i<n; i++){

fprintf(f, "%d;%s;%d\n", **Liste**[i].id, **Liste**[i].libelle, **Liste**[i].capacite);

}

fclose(f);

}

//Question 10 : Programme de test

main(){

**int** n, choix, id;

//Initialisation de la liste des salles pour les tests

/\*Salle t[30] = {{1, "S1", 20}, {2, "B2", 30}, {3, "D2", 25}}, \* a;

n = 3;\*/

**Salle** t[30], \* a;

//On suppose que l'utilisateur va commencer par le choix 1 sinon ...

do{

//Menu

puts("-------------Menu-------------");

puts("1: Lire les salles.");

puts("2: Afficher la liste des salles.");

puts("3: Supprimer une salle.");

puts("4: Afficher les salles classées par libelle.");

puts("5: Enregistrer les salles dans un fichier.");

puts("6: Quitter le programme.");

puts("Tapez votre choix :");

scanf("%d", &choix);

switch(choix){

case 1:

printf("Donner le nombre d'articles: ");

scanf("%d", &n);

**SaisirInfosSalle**(t, n);

break;

case 2:

**AfficherInfosSalles**(t,n);

break;

case 3:

printf("Entrer l'identificatuer de la salle a supprime: ");

scanf("%d", &id);

**SupprimerSalle**(id, t, &n);

break;

case 4:

**TrierSallesParLibelle**(t,n);

break;

case 5:

**Sauvegarder**(t, n);

printf("Liste sauvegardee dans le fichier Salles.txt\n");

break;

case 6:

printf("Fin du programme\n");

break;

default:

printf("Choix invalid!\n");

}

}while(choix != 6);

}  
//Source : www.exelib.net