**Solution**

**1ère année – 2ème semestre 2017**

Problème I

#include <stdio.h>

int factoriel(int n)

{

int i, fact = 1;

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

fact \*= i;

return fact;

}

float sequence(int n)

{

float Un = 1;

int i;

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

Un = Un + Un / factoriel(i);

return Un;

}

**Ou**

float sequence(int n)

{

if (n == 0) return 1;

return sequence(n - 1) + sequence(n - 1) / factoriel(n);

}

void main()

{

int n;

printf("donner n pour calculer Un");

scanf("%d", &n);

printf("\nU%d = %f", n, sequence(n));

}

Problème II

#include <stdio.h>

#define N 100

void Initialize(int T[], int n)

{

int i;

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

T[i] = -1;

}

Ou

void Initialize(int T[], int n)

{

while (n--)

T[n] = -1;

}

int Appartient(int v, int T[], int n)

{

int i, test = 0;

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

if (v == T[i])

{

test = 1;

break;

}

return test;

}

int Decompose(int n, int T[], int t)

{

int i;

for (i = 0; n != 0; i++)

{

T[i] = n % 10;

n /= 10;

}

}

void main()

{

int i, N1, N2, T1[N], T2[N], test = 1;

printf("donner 2 entier pour tester si freres ou non");

scanf("%d%d", &N1, &N2);

Initialize2(T1, N);

Initialize2(T2, N);

Decompose(N1, T1, N);

Decompose(N2, T2, N);

for (i = 0; T1[i] != -1; i++)

if (Appartient(T1[i], T2, N) == 0)

test = 0;

for (i = 0; T2[i] != -1; i++)

if (Appartient(T2[i], T1, N) == 0)

test = 0;

if (test)

printf("\n %d et %d sont freres", N1, N2);

else

printf("\n %d et %d sont non freres", N1, N2);

}

Problème III

#include <stdio.h>

#include <conio.h>

#include <string.h>

#define N 100

int estAlphabet(char c)

{

if ((c >= 'a'&&c <= 'z') || (c >= 'A'&&c <= 'Z'))

return 1;

return 0;

}

void EliminationNoAlphabet(char S[])

{

int i, j;

for (i = 0; S[i] != '\0'; i++)

if ( !estAlphabet(S[i]))

for (j = i--; S[j] != '\0'; j++)

S[j] = S[j + 1];

}

**Méthode 1 :**

void EliminationAlphabet(char S[])

{

int i, j;

for (i = 0; S[i] != '\0'; i++)

if (estAlphabet(S[i]))

for (j = i--; S[j] != '\0'; j++)

S[j] = S[j + 1];

}

void arrange(char S1[], char S2[])

{

char alpha[N], other[N], Stemp1[N], Stemp2[N];

strcpy(Stemp1, S1);

strcpy(Stemp2, S2);

EliminationNoAlphabet(S1);

EliminationNoAlphabet(S2);

strcat(S1, S2);

/\* strcat (destination, source); Elle ajoute le contenu d'une chaîne(source) à la suite d'une autre.\*/

EliminationAlphabet(Stemp1);

EliminationAlphabet(Stemp2);

strcat(Stemp1, Stemp2);

strcpy(S2, Stemp1);

}

**Méthode 2 :**

void arrange(char S1[], char S2[])

{

int i, j = 0, k = 0;

char alpha[N], other[N];

for (i = 0; S1[i] != '\0'; i++)

{

if (estAlphabet(S1[i]))

{

alpha[j] = S1[i];

j++;

// c'est meme que alpha[j++] = S1[i];

}

else

{

other[k] = S1[i];

k++;

}

}

for (i = 0; S1[i] != '\0'; i++)

{

if (estAlphabet(S1[i]))

{

alpha[j] = S1[i];

j++;

}

else

{

other[k] = S1[i];

k++;

}

}

alpha[j] = '\0';

other[k] = '\0';

for (i = 0; i <= j; i++)

S1[i] = alpha[i]; // ou avec suggestion strcpy(S1,alpha)

for (i = 0; i <= k; i++)

S2[i] = other[i];

}

**Ou**

void arrange\_helper(char Sx[], char alpha[], char other[])

{

int i, j = strlen(alpha), k = strlen(other);

for (i = 0; Sx[i] != '\0'; i++)

if (estAlphabet(Sx[i]))

alpha[j++] = Sx[i];

else

other[k++] = Sx[i];

alpha[j] = '\0';

other[k] = '\0';

}

void arrange(char S1[], char S2[])

{

char alpha[N] = { '\0' }, other[N] = { '\0' };

arrange\_helper(S1, alpha, other);

arrange\_helper(S2, alpha, other);

strcpy(S1, alpha);

strcpy (S2, other);

}