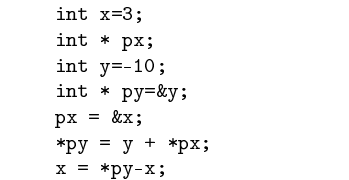
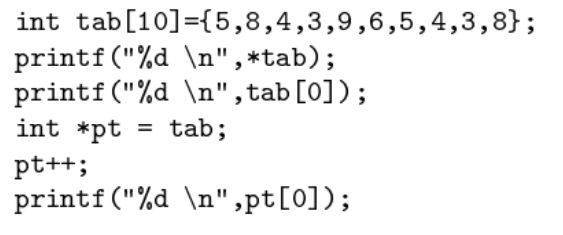
**CORRIGE TYPE**

**Exo 1 : : 2 points**

****



**Exo 2 FIBO: 5 points** (bonus)

Fonction FIB(n :entier) :entier

Bebut

si n=0 ou n=1 alors retourner 1

sinon retourner FIB(n-1) + FIB(n-2)

fin

**Exo 3 LLC: 13 points**

**Le modèle ( Opération) (2 points)**

void cree\_polynome (struct polynome \*P);

void affich\_poly (struct polynome L);

float calcul\_un\_point(struct polynome L);

void derivee ( struct polynome \*P, struct polynome \*S);

void Somme(struct polynome \*P1, struct polynome \*P2, struct polynome \*P\_somme);

void Produit (struct polynome \*P1, struct polynome \*P2, struct polynome \*P\_produit);

void permut( struct polynome \*P ,struct terme \*A , struct terme \*B ) ;

void tri\_bulle( struct polynome \*L );

**Déclaration de la struture : (3 points)**

struct terme

{

int coeff;

int pow;

struct terme \*Suiv;

} ;

struct polynome

{

struct terme \*tete;

struct terme \*queu;

int taille;

};

**Créer la structure**

struct terme \*Allouer ( )

{

return ((struct terme \*) malloc( sizeof(struct terme)));

}

**Créer polynôme :**

void cree\_polynome (struct polynome \*P)

{

struct terme \*V, \*Q;

int i;

printf("Veuillez introduire le nombre de termes du polynome :");

scanf("%d",&P->taille);

P->tete =NULL;

for (i=0; i < P->taille; i++)

{ V= Allouer ();

aff\_terme(V);

if (P->tete != NULL) Affadr(Q,V);

else (P->tete = V);

Q=V;

}

Affadr(Q,NULL);

P->queu = Q;

}

**Ajouter un élément :**

void ajouter\_polynome (struct polynome \*P,)

{

struct terme \*V, \*Q;

int i;

printf("Veuillez introduire le nombre de termes du polynome :");

scanf("%d",&P->taille);

P->tete =NULL;

for (i=0; i < P->taille; i++)

{ V= Allouer ();

aff\_terme(V);

if (P->tete != NULL) Affadr(Q,V);

else (P->tete = V);

Q=V;

}

Affadr(Q,NULL);

P->queu = Q;

}

**Supprimer un élément :**

void supprimer\_polynome (struct polynome \*P)

{ struct terme \*V, \*Q;

int i;

printf("Veuillez introduire le numéro de termes du polynome à supprimer :");

scanf("%d",&P->taille);

P->tete =NULL;

for (i=0; i < P->taille; i++)

{ V= Allouer ();

aff\_terme(V);

if (P->tete != NULL) Affadr(Q,V);

else (P->tete = V);

Q=V;

}

Affadr(Q,NULL);

P->queu = Q;

}

**Calcule de integrale : (2 points)**

void integrale ( struct polynome \*P, struct polynome \*S)

{

struct terme \*R,\*V,\*C;

int cpt=0;

S->tete=NULL;

R= P->tete;

while (R!=NULL)

{

if ((R->coeff != 0) && (R->pow !=0))

{cpt++;

V= Allouer ();

V->coeff= R->coeff /(R->pow+1);

V->pow= R->pow +1;

if (S->tete != NULL) Affadr(C,V);

else (S->tete = V);

C=V;

}

R=Suivant(R);

}

Affadr(C,NULL);

S->queu = C;

S->taille=cpt;

}

**Calcule Produit : (3 points)**

void Produit (struct polynome \*P1, struct polynome \*P2, struct polynome \*P\_produit)

{

int cpt=0;

struct terme \*P,\*Q,\*R,\*C;

printf("Polynome 1:\n");

cree\_polynome(P1);

printf("Polynome 2:\n");

cree\_polynome(P2);

P\_produit->tete=NULL;

P=P1->tete;

Q=P2->tete;

while (P !=NULL)

{

Q=P2->tete;

while (Q != NULL)

{

R=Allouer();

R->coeff= P->coeff \* Q->coeff;

R->pow = P->pow + Q->pow;

if (P\_produit->tete != NULL) Affadr(C,R);

else (P\_produit->tete=R) ;

C=R;

Q=Suivant(Q);

cpt++;

R=Suivant(R);

}

P=Suivant(P);

}

P\_produit->queu=C;

Affadr(C,NULL);

P\_produit->taille=cpt;

}

**Calcule de somme (3 points):**

void Somme(struct polynome \*P1, struct polynome \*P2, struct polynome \*P\_somme)

{

int cpt=0;

struct terme \*P,\*Q,\*R,\*C;

printf("Polynome 1:\n");

cree\_polynome(P1);

printf("Polynome 2:\n");

cree\_polynome(P2);

P\_somme->tete=NULL;

P=P1->tete;

Q=P2->tete;

while ((P != NULL) && (Q != NULL))

{

R=Allouer();

if ((P->pow) > (Q->pow))

{

R->coeff= Q->coeff;

R->pow = Q->pow;

Q= Suivant(Q);

}

else

{

if ((P->pow) < (Q->pow))

{

R->coeff= P->coeff;

R->pow = P->pow;

P= Suivant(P);

}

else

{

R->coeff= P->coeff+Q->coeff;

R->pow = P->pow;

P= Suivant(P);

Q= Suivant(Q);

}

}

if (P\_somme->tete != NULL) Affadr(C,R);

else (P\_somme->tete=R) ;

C=R;

cpt++;

R=Suivant(R);

}

while (P!=NULL)

{

R= Allouer();

R->coeff = P->coeff;

R->pow = P->pow;

if (P\_somme->tete != NULL) Affadr(C,R);

else (P\_somme->tete=R) ;

C=R;

P=Suivant(P);

R=Suivant(R);

cpt++;

}

while (Q!=NULL)

{

R= Allouer();

R->coeff = Q->coeff;

R->pow = Q->pow;

if (P\_somme->tete != NULL) Affadr(C,R);

else (P\_somme->tete=R) ;

C=R;

Q=Suivant(Q);

R=Suivant(R);

cpt++;

}

P\_somme->queu=C;

Affadr(C,NULL);

P\_somme->taille=cpt;

}

* --------------

**calcul un point (2 points )**

float calcul\_un\_point(struct polynome L)

{

struct terme \*T;

float val,s=0;

T= L.tete;

printf("Veuillez introduire la valeur pour laquelle vous voulez calculer le polynome : ");

scanf("%f",&val);

while (T!=NULL)

{

s= s+(T->coeff \* pow(val,T->pow)) ;

T= Suivant(T);

}

printf("\nLe resultat est : ");

return s;

}

**Algorithme Tri (2 points )**

void tri\_bulle( struct polynome \*L ) {

int cpt;

struct terme \*P ;

do { P = L->tete;

cpt = 0 ;

while (P != L->queu) {

if ( P->pow > ((Suivant(P))->pow)) {

permut(L,P,Suivant(P)) ;

cpt++;

}

P = Suivant(P) ;

}

} while (cpt != 0) ;

}

#endif