Questions préparatoires à l'examen No 2

Question 1 (type char)

La fonction « get_NIP( ) » retourne la valeur d'un code NIP lu un caractère à la fois. Modifiez cette fonction pour inclure le traitement des commandes « Annuler » avec la touche <ESC> (code ASCII #27) et « Correction » avec la touche <BKSPACE> (code #8). La commade « Annuler » devra arrêter immédiatement la saisie du code et retourner la valeur « 0 ». La commande « Correction » va arrêter la saisie du code courant et recommencer une nouvelle saisie de code (donc, redémarrer la somme à zéro dans « total »).

```
int  get_NIP(void){
     char ch;
     int  total, valch;
     total = 0;

     do {
          ch = getch();
          if ((ch>='0') && (ch<='9')) {
               total = total*10 + ch - '0';
               printf("*");
          }
     } while ((ch>='0') && (ch<='9'));

     return total;
}
```

Question 2 (passage par adresse)

Donnez les résultats à l'écran (valeurs de **a** et **b**) du programme suivant:

```
#include <stdio.h>

void P(int , int *);

int  a,b;

void main() {
  a=2;  b=7;
  printf("Avant P: a=%d b=%d\n", a, b);
  P(b, &a);
  printf("Après P: a=%d b=%d\n", a, b);
}

void P(int a, int *b) {
  a++;  (*b)++;
  printf("Dans  P: a=%d b=%d\n", a, *b);
}
```

Question 3 (tableaux)

Avec les déclarations suivantes dans le « main » :

```
#define  NB  100
int  tabA[NB], tabB[NB], tabC[NB];
```

Écrire la fonction « sépare( ) » qui séparera le contenu du tableau « tabA » en 2 autres tableaux « tabB » et « tabC ». Toutes les valeurs positives de « tabA » iront dans la même case de « tabB » et toutes les valeurs négatives de « tabA » iront dans la même case de « tabC ». Toutes les autres cases de « tabB » et « tabC » devont êtres 0. La fonction doit aussi retourner via 2 paramêtres-pointeurs la somme « totB » de tout les positifs, et de tout les négatifs « totC ».

```
void  separe( int tabA[], int tabB[], int tabC[], int dim,
              long *totB, long *totC );
```

Ex. pour un «dim» de 10, et tabA = `{ 2, -4, 3, 0, 6, -4, -10, 6, 2, -2 }`
nous auront les tableaux tabB = `{ 2,  0, 3, 0, 6,  0,  0, 6, 2,  0 }`
et tabC = `{ 0, -4, 0, 0, 0, -4, -10, 0, 0, -2 }`
et *totB = `19`, *totC = `-20`

Question 4 (adresses)

Qu'est-ce qui sera affiché par le programme suivant :

```
#include <stdio.h>

void F1( int p1, int p2 )
{  p1 += 2;
   p2  = p1;
}

void F2( int *p1, int *p2 )
{  *p1 += *p2;
   *p2  = *p2 / 2;
}

void F3( int p1, int *p2 )
{  p1 += *p2;
   *p2  = 3;
}

void main()
{ int  a=2,  b=2;
  F1(a, b);
  printf("a=%d b=%d\n", a, b);   /* a =_____ et  b =_____ */
  F2(&a, &b);
```

```
  printf("a=%d b=%d\n", a, b);   /* a =_____ et  b =_____ */
  F3(a, &b);
  printf("a=%d b=%d\n", a, b);   /* a =_____ et  b =_____ */
}
```

Question 5 (correction)

Ce programme est bourré **d'erreurs**, **d'oublis** ou d'étrangetés. On peut parfois en trouver plus d'une par ligne. Vous en identifiez **10** directement sur votre copie d'examen et donnez-moi un petit commentaire d'explication pour chaque erreur :

```
#include <stdio.h>

double pratique( double ) ;

void main( );
{ int  a ; b = 5 ;
  int &pti = *a ;
  int  toto[10];
  float  x
  double  y == 1 ;
  double *pt = &x ;

  scanf ( "%f" , &b ) ;
  printf ( "\n Valeur = ", b ) ;

  for ( j = 0  ;  j <= 10  ;  j++ )
     &toto [ j ] = 2( j - 1 );
  scanf ( "%lf , &y" );
  while ( x = y );
  {
     pratique( y ) = x;
     getch( );

  /* par exemple ici, on voit bien que le bloc du while n'a pas été
     fermé. Ceci ne compte pas comme une des erreurs à trouver ! */

  printf(" le programme se termine ici  " ) ;
  return  0;
}

void pratique( double x );
{ return x+2; }
```

Question 6 (getchar)

Vous devez écrire un programme complet qui lit dans un fichier de texte (par redirection) appelé "code.txt" un message codé et le retranscrit décodé caractère par caractère dans un autre fichier de texte (par redirection) appelé "résultat.txt". On présume que le nombre de caractères dans "code.txt" est inconnu. Donnez la commande de redirection utilisée.

Chaque caractère du fichier codé représente le vrai caractère incrémenté de 2. Un 'c' équivaut donc à un 'a' , un 'F' équivaut à un ' D' etc...

Exemple :

fichier codé	fichier résultat
DQPLQWT"NG"OQPFG	BONJOUR LE MONDE

Question 7 (fonctions et adresses)

Avec les déclarations suivantes :

```
void  swap(int *ptr1, int *ptr2);
void  mystere(int *ptr, int y);

int  main()
{ int  x = 5 ;
  int  y = 2 ;

  mystere (&x, y);

  printf("%d", x );
  return 0;
}

/* swap() échange les valeurs aux adresses données à « ptr1 » et « ptr2 » */
void  swap( int *ptr1, int *ptr2 )
{
  int tmp = *ptr1;
  *ptr1 = *ptr2;
  *ptr2 = tmp;
}

void mystere (int *ptr, int y)
{ int i , tot = 1 ;

  if (*ptr > y) swap( *ptr, y );

  for (i = 0 ; i < y ; i++)
    tot *= *ptr;

  ptr = tot;    /*on place la valeur de « tot » à l'adresse de « ptr » */
}
```

En présumant que « main », la fonction « swap » et les protoypes de fonctions sont tous corrects.

A) Corrigez toutes les erreurs de syntaxe que vous trouvez dans le bloc de la fonction « mystere » pour qu'elle puisse compiler correctement.

B) Donnez le résultat du « printf » dans le bloc « main ».

Question 8 (tableau)

A) Écrire une fonction " posi " qui retourne le nombre d'éléments positifs dans un tableau "tab[ ]".

```
int  posi (double tab[], int longueur);
```

B) Ajouter un paramètre "total" en référence à votre fonction "posi" qui permettera de renvoyer la somme de tous ses éléments positifs.

```
int  posi (double tab[], int longueur, double * total);
```

Question 9 (tableau)

Qu'est-ce qui sera imprimé par le programme suivant?

```
#include <stdio.h>

void main()
{ int  Tableau[3][3];
  int  I, J;

  for  (I=0; I<3; I++)
    for  (J=0; J<3; J++)
      Tableau[I][J] = I*10 + J;

  for (I=0; I<3; I++)
    printf("%3d",Tableau[2][I]);
}
```

A) **1 11 21**

B) **20 21 22**

C) **12 22 32**

D) **2 12 22**

E) Aucune de ces réponses

Question 10 (énoncés)

Choisir le fragment de programme équivalent à:

```
while (i<10) do {
   printf("%d", i++);
}
```

A)
```
for (i=0; i<10; i++)
  printf("%d", i);
```

C)
```
do{
  printf("%d", i++);
} while (i<=10);
```

B)
```
if (i<10) {
  do{
    printf("%d", i++);
  } while (i<10);
}
```

D)
```
if (i<10) {
  do{
    printf("%d", i++);
  } while (i<=10);
}
```

E) aucune de ces réponses.

Question 11 (adresses)

Donnez les résultats à l'écran du programme suivant:

```
#include <stdio.h>

void main() {
  int  x, y,
```

```
  *p=&x; *q=&y; *r;

  *p = 2;
  *q = 5 * *p;
  printf("%d %d\n", *p, *q);

  r = p;    p = q;    q = r;
  *q = x + 2;

  printf("%d %d %d\n", x, y, *r);
}
```