



PROGRAMMATION IMPÉRATIVE
PG109
JOACHIM BRUNEAU-QUEYREIX, GUILLAUME MERCIER ET NIC
VOLANSCHI

Filière : Télécom

Année : 2019

Semestre : 1

Date de l'examen : 10 Janvier 2018

Durée de l'examen : 1h

Documents autorisés ☐ sans document ☒Calculatrice autorisée ☐ non autorisée ☒

SUJET

- Merci de répondre dans les espaces prévus à cet effet.
- Le barème est indicatif.

Question 1 (1 point)

Soit le programme suivant :

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

int f (int a, int b)
{
    a = a - b;
    b = a + b;
    a = b - a;

    return 0;
}

int main(int argc, char *argv[])
{
    int x = 2;
    int y = 3;

    f(x,y);

    printf("%i %i\n",x,y);

    return 0;
}
```

Qu'affiche ce programme ?

Réponse :

Question 2 (2 points)

Écrivez un programme `moyenne` qui calcule la moyenne des nombres pris en argument sur la ligne de commande. On supposera que le programme est appelé correctement, c'est à dire que seuls des nombres sont passés en argument.

Question 3 (1 points)

Que fait la fonction suivante ?

```
int fonction(char *s)
{
    char *ptr = s;
    while( *s++ );
    return (s - ptr - 1);
}
```

Réponse :

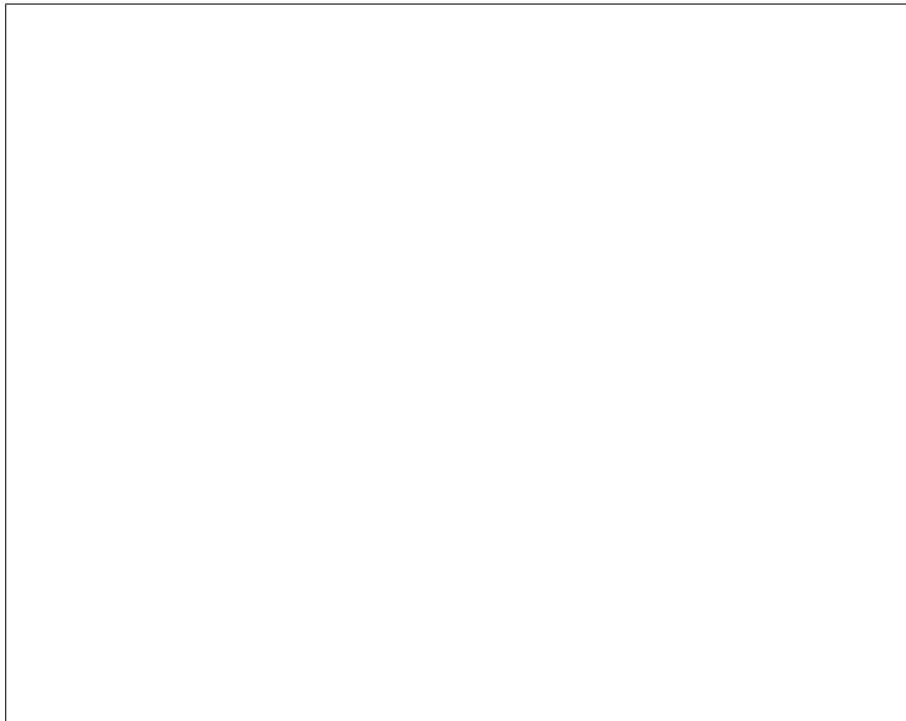
Question 4 (2 points)

Écrivez un programme (`sorted`) qui :

- stocke dans un tableau une série de nombres entiers passés en paramètres
- vérifie si cet ensemble est trié dans l'ordre croissant ou décroissant, ou non trié.

Par exemple :

```
$ ./sorted 1 3 3 5 10
increasing
$ ./sorted 12 10 5 1 -3
decreasing
$ ./sorted 1 4 6 5 9
unsorted
$ ./sorted 1 1 1 1 1
increasing
```



Question 5 (2 points)

Soit le programme suivant :

```

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

typedef struct element {
    char clef;
    int  valeur;
} elem_t ;

int main(int argc, char *argv[])
{
    elem_t *el = NULL;
    char key = 'a';
    int  val = 12;

    el = alloc_and_init(key,val);

    /* reste du code */

    exit(EXIT_SUCCESS);
}

```

Écrivez la fonction `alloc_and_init` qui alloue une nouvelle structure et l'initialise avec les valeurs prises en argument.

```

elem_t *alloc_and_init(char key, int value)
{

}

```

Question 6 (3 points)

Écrivez un programme qui vérifie si une chaîne de caractères passée en argument est un palindrome ou non (un palindrome est un mot qui se lit de la même façon de gauche à droite ou de droite à gauche, ex : été).

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

int main(int argc, char *argv[])
{

exit(EXIT_SUCCESS);
}
```

Question 7 (2 points)

Écrivez une fonction `int pi(unsigned int n)` qui calcule et affiche une approximation de π avec le n -ième terme de la suite de Leibniz-Grégory. Cette suite est définie de la façon suivante :

- $u_0 = 4$
- $u_n = u_{n-1} + \frac{4 \times (-1)^n}{2n+1}$ avec $n \geq 1$

C'est à dire : $\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \dots$

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

int pi(unsigned int n)
{

}

int main(int argc, char *argv[])
{

}

exit(EXIT_SUCCESS);
}
```

Question 8 (3 points)

Soit le programme suivant :

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

#define max(a,b) ((a) > (b) ? (a) : (b))

int main(int argc, char *argv[])
{
    int t[] = {1, 2, 3};
    int x = 1, y = 2;
    int i = x-- + --y;

    printf("%d\n", i);          // q1
    printf("%d\n", t[i--]);    // q2
    printf("%d\n", t[--i]);    // q3
    printf("%d\n", t[1]);      // q4
    printf("%d\n", t[0]++);    // q5
    printf("%d\n", --t[0]);    // q6
}
```

Quel est l'affichage de ce programme ?

```
q1 :
q2 :
q3 :
q4 :
q5 :
q6 :
```

Question 9 (1 points)

Soit le programme suivant :

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    int tab[10] = {0,2,4,6,8,10,12,14,16,18};
    int x = 0;

    x = tab[3] + *(tab + 5) ;

    fprintf(stdout,"valeur de x : %d \n",x);

    exit(EXIT_SUCCESS);
}
```

Quel est l'affichage de ce programme ?

Réponse :

Question 10 (1 points)

Soient les déclarations suivantes :

```
struct bidule{
    int  machin;
    char truc;
};

struct bidule toto;
```

Quelle est la taille de la variable `toto` (entourez la bonne réponse) ?

1. au plus la somme `sizeof(int) + sizeof(char)` ?
2. le maximum entre `sizeof(int)` et `sizeof(char)` ?
3. exactement la somme `sizeof(int) + sizeof(char)` ?
4. au moins la somme `sizeof(int) + sizeof(char)` ?

Question 11 (2 points)

Soit la déclaration suivante :

```
struct bidule{
    int  indice;
    char *mot;
    struct bidule *suivant;
};
```

Écrivez un programme qui

- récupère des chaînes passées en argument
- alloue une nouvelle structure telle que déclarée ci-dessus
- copie le mot passé en argument dans la structure et affecte un indice au mot
- effectue le chaînage avec les éléments précédents (avec le pointeur `suivant`).


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

struct bidule{
    int  indice;
    char *mot;
    struct bidule *suivant;
};

int main(int argc, char *argv[]){

    exit(EXIT_SUCCESS);
}
```