PRÉNOM: Raquel

# Programmation C

Révisions

## Question 1:

Soit le programme C suivant:

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

int main() {
  int x=1, y=2;
  f(x, &y); printf("%d %d\n", x, y);
}
```

f( 1, &y ) {
y = 2 * 2; printf( 1, 4);
a = 2+4; printf( 6, 5);
y = 5 ;
}
x = 1
y = 5

Indiquez les valeurs imprimées sur la sortie standard.

2 4
6 5
1 5

## Question 2:

Écrire la fonction

```
char *strdup(const char *s);
```

qui crée une copie de la chaîne de caractères s. Cette fonction renvoie un pointeur sur la copie si celle-ci a pu se faire et NULL dans le cas contraire.

(1)
```
char *strdup(const char *s) {
  char *d = malloc (strlen(s) +1); //space for length plus nul
  if(d==NULL) return NULL; //no memory
  strcpy(d, s);            //copy the characters
  return d;                //return the new string
}
```

(2)
```
char *strdup(const char * s) {
  if(strlen(s) == 0) return NULL;
  char * str_copy = malloc(sizeof(s));
  char * ptr_temp_s = s;
  char* ptr_temp_copy = str_copy;
  while(*ptr_temp_s)*(ptr_temp_copy++) = *(ptr_temp_s++)
  return str_copy;}
```

**Question 3:**

Écrire le programme

```
cmp fich1 fich2
```

qui compare les fichiers `fichier1` et `fichier2`. Si les deux fichiers sont identiques le programme n'affiche rien, sinon il affiche le caractère et la ligne où la différence a été trouvée avant de s'arrêter. Pour simplifier, on suppose ici que le programme est bien appelé avec deux fichiers et que ceux-ci existent et sont lisibles.

```
int cmp(char *file1name, char*file2-name){
    FILE* file1, *file2;
    file1 = fopen(file1-name,"r");
    file2 = fopen(file2-name,"r");
    int c1,c2;
    while((c1 = fgetc(file1)) != EOF && (c2 = fgetc(file2)) != EOF)
    { if (c1 != c2) return 0; }
    return 1;
}

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

    if (strcmp(argv[1], "cmp") == 0){
        check_argument(argc, argv);
        printf("is diff ? %d \n", cmp(argv[2], argv[3]));
    } else{
        puts(readline(argv));
    }
    return 0;
}
```

**Question 4:**

Ajouter les contrôles nécessaires à une bonne exécution du programme `cmp` précédent. On ne veut ici que la partie qui contrôle les paramètres. Inutile de recopier ici votre réponse à la question précédente.

```
int check-argument(int argc, cons char ** pString){
    if (argc <4 && argc >4 )return 0;
    if (strcmp(pString[1], "cmp) != 0) return 0;

    FILE* file1, * file2,
    file1 = fopen(pString[2], "r");
    file2 = fopen(pString[3], "r");
    if(!file1 || !file2) return 0;
     return 1;
}
```

**Question 5:**

Écrire le programme

```
head [ -n ] [ fichier ]
```

qui copie les n premières lignes de `fichier` sur la sortie standard. Si `fichier` n'est pas spécifié, `head` prend ses lignes sur l'entrée standard. La valeur par défaut pour n est de 10 lignes.

On rappelle que la fonction `int atoi(const char *str);` peut être utilisée pour convertir la chaîne de caractères `str` en entier.

```
#define BUFFER_MAX_LENGTH 1024

char* read_line(char const * argv[]){
    int nline = atoi(argv[3]);
    File *file = fopen(argv[4], "r");
    if(!file) return NULL;

    char copy[BUFFER_MAX_LENGTH];
    char line[BUFFER_MAX_LENGTH];

    while(fgets(line, sizeof(line), file) && nline>0) {
        strcat(copy, line);
        nline--;
    }
    return copy;
}
```