TD2 : MPC.

3. En utilisant la fonction `strend()`, proposez une fonction qui renvoie le nombre de caractères d'une chaîne donnée - le caractère `'\0'` final non compris :

```c
int mstrlen(const char *str);
```

Pour information. Cette fonction `mstrlen()` est similaire à la fonction `strlen()` fournie par la bibliothèque standard `string.h`.

```c
int strlen(const char *str){
char *tmp
tmp = str
Return strend(tmp)-str;
}
```

4. La bibliothèque `string.h` fournie également les fonctions suivantes :

```c
/* recopie le contenu de src dans dest renvoie dest */
char *strcpy(char *dest, const char *src);

/* recopie src à la fin de dest (concat) renvoie dest */
char *strcat(char *dest, const char *src);
```

Proposez des fonctions `mstrcpy()` et `mstrcat()`, réécritures de ces fonctions.

```c
char *strcpy(char *dest, const char *src){
int i = 0;
for(; src[i++];)  dest[i] = char[i]
}
```

Exercice 23. [Allocation dynamique] Soit le programme *incorrect* suivant:

```c
/* Renvoie un tableau à valeurs:
   {0, 1, 2, ..., 9} */
int *creer_sequence_10(){
  int t[10];
  int i;
  for (i=0; i<10; i++)
    t[i] = i;
  return t;
}
```

:c libéré au Return :c

```c
int main(void){
  int *tab;
  int i;
  tab = creer_sequence_10();
  for (i=0; i<10; i++)
    printf("%d\n", tab[i]);
}
```

free(tab);
Return 0;
}

1. Pourquoi ce programme est-il incorrect ?

2. Proposez une correction de ce programme utilisant `malloc()` et `free()`.

```c
int *t = (int*) malloc(10 * sizeof(int))
```

Exercice 24. [Allocation et libération de structures.]
Soit les déclarations suivantes:

```c
struct point_s{
  int p_x;
  int p_y;
};
```

```c
struct point_s *new_point(int x, int y);
void free_point(struct point_s *p);
```

La fonction `new_point()` alloue et renvoie l'adresse d'une nouvelle structure de type `struct point_s`, dont les champs sont initialisés avec les valeurs des paramètres `x` et `y`. La fonction `free_point()` se charge de libérer la mémoire allouée correspondante à `p`.

1. Donnez une définition de la fonction `new_point()`.

```c
struct point_s* new_point(int x
                          int y) {
struct point_s * res = malloc(sizeof(point_s))
Res -> x = x
Res -> y = y
Return Res. }
```

2. Donnez une définition de la fonction `free_point()`.

```c
free_point(struct point_s *p) {
    free(p);
}
```