Université de Nice-Sophia Antipolis
Polytech'Sophia

Université de Xidian
2025–2026

Examen de Langage C

Durée : 1h30
Aucun document autorisé – Mobiles interdits

Notez que les affirmations (antécédents, conséquents, rôles, et invariants) dans vos codes C entreront pour partie dans la note finale.

▶ 1. En utilisant uniquement la notation pointeur, et SANS UTILISER de fonctions de `string.h`, écrivez la fonction `strlen` qui renvoie la longueur d'une chaîne de caractères passée en paramètre.

Question sur 3 pts

```c
/*
 * antécédent : s!=NULL
 * rôle : renvoie la longueur de la chaîne de caractères s
 */
int strlen(const char *s) {
  int i=0;
  while (*s++) i++;
  return i;
}
```

.....

▶ 2. En utilisant uniquement la notation pointeur, et SANS UTILISER de fonctions de `string.h`, écrivez la fonction `replaceDigits` qui remplace dans une chaîne de caractères `s` tous les chiffres par un caractère `c`. Cette fonction ne possède que les deux paramètres `s` et `c`. La fonction `replaceDigits` renvoie la chaîne de caractères `s` modifiée.

Question sur 4 pts

```c
/*
 * antécédent : s!=NULL, c caractère de remplacement
 * rôle : renvoie la chaîne s dans laquelle les chiffres
 *         sont remplacés par la caractère c
 */
char *replaceDigits(char *s, const char c) {
  char *p = s;
  while (*s) {
    if (isdigit(*s))
      // remplacer le chiffre *s par c
      *s =c;
    //
    s++;
  }
  return p;
}
```


.....

▶ 3. Écrivez la fonction `main` dans laquelle vous testerez votre fonction `replaceDigits`.

Question sur 4 pts

```c
int main(void) {
  char s[] = "123ccc876";
  printf("(%s)\n", replaceDigits(s, '-'));
  return EXIT_SUCCESS;
}
```

.....

▶ 4. À l'aide d'un **typedef**, déclarez le type `Complexe`, une structure, pour représenter des nombres complexes sous forme cartésienne. La structure `Complexe` possède les deux champs `préel` et `pimg` de type **double**.

Question sur 2 pts

```c
typedef struct {
  double préel, pimg;
} Complexe;
```

.....

▶ 5. Écrivez la fonction `add` qui renvoie la somme de deux `Complexe` `c1` et `c2` passés en paramètres.

Question sur 2 pts

```c
/*
 * rôle : renvoie c1+c2
 */
Complexe add(const Complexe c1, const Complexe c2) {
  return (Complexe) {c1.préel+c2.préel, c1.pimg+c2.pimg};
}
```

.....

▶ 6. Écrivez le fragment de code qui déclare et initialise deux complexes `c1` et `c2` et qui calcule leur somme.

Question sur 1 pt

```c
Complexe c1 = {3.5, -2.1};
Complexe c2 = {0.0, 1.0};
//
Complexe c3 = add(c1,c2);
```

.....

▶ 7. En utilisant uniquement la notation pointeur, écrivez la fonction `sommeComplexes` qui prend en

paramètre un tableau `t` de pointeurs sur `Complexe` et son nombre d'éléments `n`. Cette fonction renvoie la somme de tous les complexes pointés contenus dans le tableau. Utilisez votre fonction `add`.

Question sur 4 pts

```
/*
 * antécédent : n>0, nb d'élements de t
 * rôle : renvoie la somme des complexes pointés par tous les
 *         éléments de t
 */
Complexe sommeComplexes(Complexe **t, const int n) {
  Complexe c = **t;
  for (int i=1; i<n; i++)
    c = add(c, *(*t++));
  //
  return c;
}
```

.....