Nom : **Prénom :**

Université de Nice-Sophia Antipolis — POLYTECH
ELEC3 — 2016–2017

Examen de Langage C

Durée : 1h
Aucun document autorisé
Mobiles interdits Documents non autorisés

Note : la qualité des commentaires, avec notamment la présence d'affirmations significatives et d'invariant, ainsi que les noms donnés aux variables, et la bonne indentation rentreront pour une part importante dans l'appréciation du travail.

▶ 1. Soit le programme C suivant :

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

void p(char *t, int *i) {
  t[++*i] = 'a';
}

int main(void) {
  char a[] = { 't', 'z'};
  int b = 0;
  p(a,&b);
  printf("%c - %c - %d\n", *a, *(a+1), b);
  return EXIT_SUCCESS;
}
```

---

1. Est-ce que `gcc` compile ce programme sans erreur ?
2. Si oui, qu'est-ce qu'est écrit sur la sortie standard ?

---

1. oui
2. `t - a - 1`

.....

▶ 2. En utilisant la notation de pointeur, écrivez en C la fonction `remplacerChiffres` qui prend en paramètre une chaîne de caractères, et qui remplace chaque chiffre par le caractère `'_'` dans celle-ci. La fonction renvoie un pointeur sur le 1er caractère de la chaîne modifiée. Vous pourrez utiliser la fonction `isdigit`, à l'exclusion de toutes les autres.

```
/*
 * Rôle : renvoie un pointeur sur le premier caractère de
 *        la chaîne s dans laquelle les chiffres ont été remplacés
 *        par le caractère '-'
 */
char * remplacerChiffres(char  *s) {
  char * p = s;
  while (*s) {
    if (isdigit(*s))
      *s = '-';
    s++;
  }
  return p;
}
```

.....

▶ 3. Écrivez en C une fonction `main` qui donne un exemple d'utilisation de votre fonction `remplacerChiffres`.

```
int main(void) {
  char t[] = "hello 06 qssdf 98";
  printf("s = %s\n", remplacerChiffres(t));
  return EXIT_SUCCESS;
}
```

.....

On veut représenter une suite d'entiers par une structure de liste dynamique doublement chaînée. Pour cela, les éléments de type `noeud` seront chaînés entre eux par deux liens, un pointeur qui pointe sur l'élément suivant et un autre qui pointe sur l'élément précédent. On considérera que chaque élément mémorise une valeur entière.

▶ 4. Complétez la déclaration de type suivante pour qu'elle soit conforme aux indications précédentes.

```
struct noeud { ..... };
```

```
struct noeud {
  int elt;
  struct noeud *precedent;
  struct noeud *suivant;
}
```

.....

Les éléments d'une liste pourront être accessibles par le début (le premier nœud) ou par la fin (le dernier nœud).

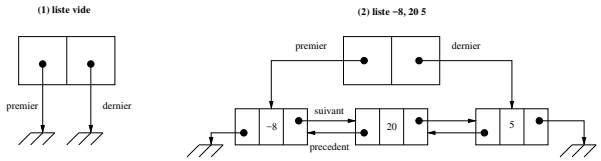
▶ 5. Complétez la déclaration suivante pour définir une liste conforme aux indications précédentes.

```
typedef  ..... Liste;
```

```
typedef struct {
  struct noeud *premier;
  struct noeud *dernier;
} Liste;
```

.....

▶ 6. Selon la structure précédente, dessinez précisément : (1) une liste vide, (2) une liste contenant les trois entiers -8, 20, et 5.



.....

▶ 7. Écrivez en C la procédure `allerRetour` qui prend une `Liste` en paramètre, et qui effectue son parcours aller et retour (i.e. du premier au dernier, et du dernier au premier).

```
/*
 * Rôle : parcourt la liste du premier au dernier élément
 *        et du dernier au premier
 */
void allerRetour(Liste l) {
  struct noeud *p = l.premier;
  // l'aller
  while (p!=NULL) p=p->suivant;
  // le retour
  p = l.dernier;
  while  (p!=NULL) p=p->precedent;
}
```

.....

▶ 8. Écrivez en C la procédure `ajouterALaFin` qui ajoute dans une liste `l`, après le dernier élément, un nouvel entier `x`. La liste et l'entier sont les deux seuls paramètres de la procédure.

```
/*
 * Rôle : ajoute en fin de liste *l  l'entier x
 */
void ajouterALaFin(Liste *l, int x) {
  // créer le nouveau nœud
  struct noeud *p = malloc(sizeof (struct noeud *));
  p->elt = x;
  p->suivant = NULL;
  // insérer le nouveau nœud
  if (l->premier==NULL) {
```



```
    // la liste est vide
    p->precedent = NULL;
    l->premier=p;
  }
  else {
    // cas général
    p->precedent = l->dernier;
    l->dernier->suivant=p;
  }
  // accrocher le nouveau nœud à l->dernier
  l->dernier=p;
}
```

.....