Licence 2, Partiel LC4, Développement en C
17 mars 2007, durée 2h

1 Question de cours

▶ **Exercice 1**

```
CC=gcc
CFLAGS=-Wall -pedantic

OBJECTS=fonctions.o

gestion: gestion.c $(OBJECTS) fonctions.h
	$(CC) $(CFLAGS) -o gestion gestion.c $(OBJECTS)

fonctions.o: fonctions.c fonctions.h
	$(CC) $(CFLAGS) -c -o fonctions.o fontions.c

clean:
	rm -rf *~ \#*

cleanall:
	rm -rf *~ \#* $(OBJECTS) gestion
```

2 Pointeurs

▶ **Exercice 2**

```
int *concat(int* t1, int n1, int* t2, int n2) {
  int * tmp;
  int i;
  if ((tmp = realloc(t1, (n1+n2) * sizeof(int))) == NULL)
    return NULL;
  t1 = tmp;
  for(i = n1; i < (n1+n2); i++)
    t1[i] = t2[i-n1];
  free(t2);
  return(t1);
}
```

▶ **Exercice 3**

```
E1 = 3
E2 = 14
E3 = 3
E4 = 7
E5 = 5
E6 = 9
E7 = 12
```

3 Boucles et décalages

▶ **Exercice 4**

```
int position(unsigned int x){
  int i, k;
  for(i=0, k=-1; x != 0; i++){
```

```
    if ( x & 1 ) /* si le bit de poids faible est 1 */
      k = i;     /* memoriser l'indice i dans k    */
    x >>= 1;     /* decalage d'un bit a droite */
  }
  return k;
}
```

4 Files

► **Exercice 5**

```
file_t *alloue_file(size_t capacite) {
    file_t *file;
    if (!capacite)
        return NULL;
    if ((file = malloc(sizeof(file_t))) == NULL)
        return NULL;
    if ((file->elements = malloc(capacite * sizeof(int))) == NULL) {
        free(file);
        return NULL;
    }
    file->debut = file->fin = 0;
    file->capacite = capacite;
    return file;
}

void libere_file(file_t *file) {
    free(file->elements);
    free(file);
}
```

► **Exercice 6**

```
int est_vide(file_t *file) {
  return (file->debut == file->fin);
}
size_t taille(file_t *file) {
  if (file->fin < file->debut)
    return ((file->capacite - file->debut) + file->fin);
  else /* file->fin >= file->debut */
    return (file->fin - file->debut);
}

int est_pleine(file_t *file) {
  return taille(file) == (file->capacite - 1);
}
```

► **Exercice 7**

```
int enfile(file_t *file, int n) {
  size_t i;
  int *t;
  if (est_pleine(file)) {
```

```
    /* la file est pleine, on augmente sa capacite */
    if ((t = realloc(file->elements,
                     2 * file->capacite * sizeof(int))) == NULL)
      return n+1;
    file->elements = t;
    if (file->fin < file->debut) {
      /* on deplace les elements de la file pour occuper des cases consecutives */
      for (i = 0; i < file->fin; i++)
        file->elements[file->capacite + i] = file->elements[i];
      file->fin += file->capacite; /* on met a jour file->fin */
    }
    file->capacite *= 2; /* on met a jour file->capacite */
  }
  file->elements[file->fin] = n; /* on enfile n */
  file->fin++; /* on met a jour file->fin */
  if (file->fin == file->capacite)
    file->fin = 0;
  return n;
}

int defile(file_t *file) {
  int n;
  assert( ! est_vide(file) );
  n=file->elements[file->debut];
  file->debut++;
  if (file->debut == file->capacite)
    file->debut = 0;
  return n;
}
```