Structures de données, tableaux et listes

1 Structure de données

2 Tableaux

Implémentation en Ocaml Mettre des trous

OCaml
```
type 'a rarray = {mutable taille : int; mutable contenu : 'a array};;
let creer n v = {taille = n; contenu = Array.make n v};;
let acces t i =  if i < t.taille then t.contenu.(i)
               else failwith "index out of range";;
let modifie t i v = if i < t.taille then t.contenu.(i) <- v
                else failwith "index out of range";;
let taille t = t.taille;;
let change_taille t m = let u = Array.make m t.contenu.(0) in
  for i = 0 to (min t.taille m)-1 do u.(i) <- t.contenu.(i) done;
  t.contenu <- u;
  t.taille <- m;;
```

3 Listes

3.1 Listes chainées

Implémentation en Ocaml

```
type 'a listechainee = Nil | Cons of 'a * 'a listechainee;;
```

Implémentation en C Mettre des trous

```
typedef struct maillon {int valeur; struct maillon* suivant;} maillon;
typedef struct listechainee {maillon* premier;} listechainee;
```

Implémentation des primitives de la liste en utilisant des maillons chaînés

1. créer une liste vide

C
```
listechainee* creer(){
  listechainee* l = malloc(sizeof(listechainee));
  l->premier = NULL;
  return l;}
```

2. tester si la liste est vide

C
```
bool est_vide(listechainee* l) {
  return l->premier == NULL;}
```

3. renvoyer la valeur de tête :

C
```
int tete(listechainee *l) {
  assert(!est_vide(l));
  return l->premier->valeur;}
```

4. renvoyer la queue :

```c
listechainee* queue(listechainee* l) {
    /* Cette version de la fonction queue
    ne renvoie pas une queue indépendante de l*/
    assert(!est_vide(l));
    l->premier = l->premier->suivant;
    return l;}
```

5. ajouter un élément

```c
void ajouter_en_tete(int x, listechainee* l) {
  maillon* m = malloc(sizeof(maillon));
  m->valeur = x;
  m->suivant = l->premier;
  l->premier = m;}
```

6. retirer l'élément de tête

```c
void retire_tete(listechainee* l) {
  assert(!est_vide(l));
  maillon* l2 = l->premier;
  l->premier = l->premier->suivant;
  free(l2);}
```

7. Désallouer la liste

```c
void detruire(listechainee* l) {
  if (l->premier != NULL) {maillon* m=l->premier;
                           l->premier = m->suivant;
                           free(m);
                           detruire(l);}
  else{free(l);}}
```

3.2 Opération sur les listes