Segon examen parcial PRO2 Durada: 2h 30m 9/1/2017

Problema 1 (5 punts)

Donada una cua de parells d'enters on el primer element de cada parell és l'identificador d'un usuari i el segon element és el temps estimat de la gestió que vol realitzar, heu d'implementar una operació que reparteixi de manera justa els usuaris de la cua original en dues cues: l'original i una de nova. Les dues cues que en resultin han de satisfer les propietats següents:

1. Cap usuari ha d'esperar més en la cua a la qual ha estat assignat que un altre usuari que estigués situat darrere seu en la cua original.
2. Tots els usuaris han d'esperar el mínim temps possible.
3. Si un usuari pot estar assignat a ambdues cues satisfent les propietats anteriors, serà assignat a la cua original.

A continuació donem la implementació del tipus de dades `Cua` amb una petita modificació que permet simplificar la implementació de l'operació de distribució de cues de parells d'enters, és a dir, de *(usuari, temps)*.

```
class Cua {
  private:
    struct node {
        int usuari;
        int temps;
        node* seguent;
    };
    int longitud;
    node* primer;
    node* ultim;
};
```

Concretament, heu d'implementar l'acció `distribucio` especificada a continuació. No es poden utilitzar les operacions públiques de la classe `Cua`.

```
void distribucio (Cua& c);
// Pre: c = C i la cua paràmetre implícit és buida.
// Post: Els elements de C estan distribuïts de manera justa entre
// les cues c i paràmetre implícit.
```

Donem tot seguit un exemple de distribució justa de cues. Representem els elements de la cua com a parells d'enters, on el primer element del parell és l'identificador d'un usuari i el segon és el temps estimat de la gestió que vol realitzar. La cua `q1` abans de realitzar la crida a l'operació `distribucio` és (el primer element de la cua és el de l'esquerra):

$$(3,2) \to (6,3) \to (2,5) \to (11,1) \to (8,4) \to (5,3) \to (9,2) \to (1,3) \to (7,4) \to (15,2) \to (4,3)$$

Si `q2` és una variable de tipus `Cua` que conté una cua buida del tipus descrit anteriorment i fem la crida `q2.distribucio(q1)`, el valor de `q1` després de la crida serà

$$(3,2) \to (2,5) \to (5,3) \to (1,3) \to (15,2)$$

mentre que el valor de `q2` després de la crida serà

$$(6,3) \to (11,1) \to (8,4) \to (9,2) \to (7,4) \to (4,3)$$

Problema 2 (5 punts)

Volem utilitzar *arbres generals* de `string` per representar expressions aritmètiques del llenguatge de programació Lisp. Les expressions que considerem només poden contenir strings corresponents a nombres enters, als operadors binaris $+, -, *, /, <, ==$, a l'operador unari `abs` (que retorna el valor absolut d'un enter) i a la instrucció de control `if`. L'operador `/` representa la divisió entera. Recordeu que n/d està definit per a qualsevol parell d'enters n, d tals que $d \neq 0$. La instrucció de control `if` té tres arguments; la seva sintaxi és `(if exp_bool inst_1 inst_2)` i s'interpreta de la manera següent: si el resultat d'avaluar `exp_bool` és cert, s'avalua `inst_1` i se'n retorna el resultat; altrament, s'avalua `inst_2` i se'n retorna el resultat. Si el resultat d'avaluar `exp_bool` no és un valor booleà (`1` o `0`), es considera indefinit.

Un *arbre d'expressió* és un tipus particular d'*arbre general* instanciat per a valors de tipus `string` que permet representar expressions aritmètiques. Si volem definir una variable `a` que emmagatzemi un *arbre d'expressió*, l'hem de declarar mitjançant la instrucció `ArbreGen<string> a;`. Concretament, un *arbre d'expressió* és un *arbre general* instanciat per a valors de tipus `string` que satisfà la definició següent:

1. L'arbre d'expressió associat a l'expressió buida és l'arbre buit.
2. L'arbre d'expressió associat a l'expressió formada per un `string` *e* que representa un nombre enter és un arbre amb arrel *e* i sense fills.
3. L'arbre d'expressió associat a una expressió de la forma `(op_n arg_1 ...arg_n)` és un arbre amb arrel `op_n` i amb `n` fills, on el fill *i*-èssim és l'arbre d'expressió associat a l'argument *i*-èssim `arg_i`.

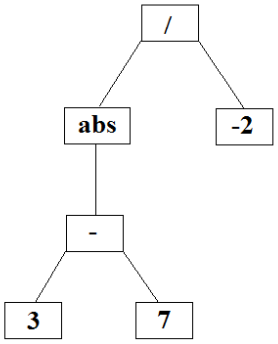
Donem tot seguit la implementació de la classe `ArbreGen`.

```
template <class T> class ArbreGen {

private:

  struct node {
    T info;
    vector<node*> seg;
  };

  node* primer_node;

     ... // especificació i implementació d'operacions privades

  public:
     ... // especificació i implementació d'operacions públiques
};
```

Una expressió aritmètica representada mitjançant un arbre d'expressió pot quedar indefinida perquè:

- conté operadors o instruccions no contemplats en la definició del primer paràgraf de l'enunciat d'aquest problema,
- un operador o instrucció que requereix n arguments s'aplica a un nombre d'arguments diferent de n, o bé
- un operador o instrucció se aplica a arguments per als quals no està definit.

Per exemple, l'expressió `(/ (abs (- 3 7)) -2)` es pot representar fent servir l'arbre d'expressió següent:



Feu un disseny recursiu de l'acció `avalua` que provi d'avaluar l'expressió aritmètica representada pel paràmetre implícit (p.i.) i, en cas que aquesta expressió aritmètica estigui definida, retorni el seu resultat a través del paràmetre de sortida `result`.

void *avalua*(**bool**& *def*, **int**& *result*) **const**;
// Pre: El p.i. és un arbre d'expressió que representa una expressió aritmètica
// ben parentitzada i no buida.
// Post: Si l'expressió aritmètica associada al p.i. està definida, assigna el valor
// `true` al paràmetre `def` i el resultat d'avaluar el p.i. al paràmetre `result`;
// altrament, assigna el valor `false` al paràmetre `def`.

En la implementació d'`avalua` podeu fer servir (sense haver d'incloure la seva definició) l'acció següent de conversió de `string` a `int` (nombre enter), que està definida en C++ amb un altre nom.

void *string_a_enter* (**const** *string*& *s*, **bool**& *es_enter*, **int**& *result*);
// Pre: cert
// Post: Si `s` representa un nombre enter, s'assigna `true` al paràmetre de sortida
// `es_enter` i el nombre enter que representa `s` al paràmetre de sortida `result`;
// altrament, s'assigna `false` al paràmetre de sortida `es_enter`.

Escriviu la capçalera, precondició, postcondició i implementació de l'acció d'immersió. Implementeu l'acció `avalua` utilitzant l'acció d'immersió. No es poden fer servir operacions públiques de la classe `ArbreGen`. Es valorarà l'eficiència de la solució.

Per exemple, si *a* és una variable de tipus `ArbreGen` instanciat en `string` que representa l'expressió aritmètica `(/ (abs (- 3 7)) -2)` mitjançant un arbre d'expressió, la crida

a.*avalua*(*def*, *res*);

assignarà a la variable booleana `def` el valor `true` i a la variable entera `res` el valor -2.

De la mateixa manera, si *b* és una variable de tipus `ArbreGen` instanciat en `string` que representa l'expressió aritmètica `(+ (/ 2 (* 0 1)) 7)` mitjançant un arbre d'expressió, la crida

b.*avalua*(*def*, *res*);

assignarà a la variable `def` el valor `false` i no assignarà cap valor a la variable entera `res`.