Longitud de segments constants als extrems d'una llista. X41684_ca

Escriviu un programa que simuli el manteniment d'una llista d'enters, a base de llegir instruccions que la van actualitzant, i instruccions que la van consultant. La llista d'enters se suposa inicialment buida, i les instruccions son dels següents tipus: afegir elements al principi o final de la llista, treure elements del principi o final de la llista, o consultar quin és el major nombre d'elements iguals començant des del principi o final de la llista.
És obligatori utilitzar només el constructor de tipus `list` com a mecanisme d'emmagatzemament massiu de dades. En particular, no es pot usar ni `vector`, ni `stack`, ni `queue`. Podeu declarar tants `list` com volgueu i del tipus que volgueu (que no sigui cap altre mecanisme d'emmagatzemament massiu), i podeu usar iteradors sobre llistes.

Entrada

La entrada consisteix en un nombre arbitrari de linies, cadascuna amb una instrucció. Les instruccions poden ser de la següent forma:

```
push_front x
push_back x
pop_front
pop_back
max_equal_left
max_equal_right
```

Sortida

Les instruccions `pop_front` i `pop_back` poden escriure `error` a la sortida, i les instruccions `max_equal_left` i `max_equal_right` escriuran un valor a la sortida. Se sobreentén que la llista que simulem està inicialment buida, i que l'efecte de cada instrucció és el següent:

- `push_front x` afegeix l'enter `x` al principi de la llista.
- `push_back x` afegeix l'enter `x` al final de la llista.
- `pop_front` elimina l'element que es troba al principi de la llista. Si la llista és buida ha d'escriure `error` i salt de linia.
- `pop_back` elimina l'element que es troba al final de la llista. Si la llista és buida ha d'escriure `error` i salt de linia.
- `max_equal_left` escriu quin és el màxim nombre d'elements iguals i consecutius que es troben començant des de la part inicial de la llista.
- `max_equal_right` escriu quin és el màxim nombre d'elements iguals i consecutius que es troben començant des de la part final de la llista.

Exemple d'entrada 1

```
max_equal_left
max_equal_right
pop_front
pop_back
push_front 0
max_equal_left
```

```
max_equal_right
push_back 3
push_back 0
max_equal_left
max_equal_right
pop_back
push_back 3
push_front 0
max_equal_left
max_equal_right
push_front 3
push_back 3
max_equal_left
max_equal_right
push_back 3
max_equal_left
max_equal_right
push_back 0
max_equal_left
max_equal_right
push_front 5
push_back 6
push_back 6
push_back 0
max_equal_left
max_equal_right
```

Exemple de sortida 1

```
0
0
error
error
1
1
1
1
2
2
1
3
1
4
1
1
1
1
```

Exemple d'entrada 2

```
push_front 0
pop_back
push_back 0
max_equal_left
push_front 0
push_front 3
max_equal_left
max_equal_right
push_front 3
pop_front
push_front 3
push_front 3
push_back -1
max_equal_right
max_equal_right
max_equal_right
push_front 2
push_front 2
max_equal_right
push_front -2
push_front -1
push_front 2
push_back 2
push_front 2
push_front -2
push_front -2
push_back -2
pop_front
pop_back
push_back -2
pop_front
```

```
push_front 3
push_back 0
push_back 0
max_equal_left
push_front 2
push_front 2
push_front 2
pop_back
push_back 2
pop_back
push_front 0
max_equal_right
push_front 0
push_back -1
pop_front
push_front -1
push_back -1
push_front -2
push_back -2
```

Exemple de sortida 2

```
1
1
2
1
```

```
1
1
1
1
1
```

Observació

Per tal de superar els jocs de proves públics i aconseguir una nota raonable, podeu fer una implementació senzilla mantenint la llista representada en un `list<int>`, i a on el càlcul de `max_equal_left` i `max_equal_right` tingui cost lineal i consisteixi en recorrer la llista des del principi o final mentre hi trobem elements idèntics. De fet, us recomanem que no us lieu i seguiu aquest enfoc.
Però els jocs de proves privats són grans. Per tal d'aconseguir superar-los tots i obtenir així la màxima nota, convindrà trobar un enfoc més eficient.

Informació del problema

Autor : PRO1
Generació : 2022-09-13 23:07:24

© *Jutge.org*, 2006–2022.
https://jutge.org