UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

Exercicis amb el Jutge

En aquesta sessió resoldrem diversos exercicis fent servir la plataforma Jutge. Primer els resoldrem fent servir el nostre editor i compilador habituals i, un cop provats, els enviarem al Jutge. Podeu provar els jocs de prova públics que proporciona el Jutge en el vostre ordinador si els descarregueu cliclant la icona "Get problem files as ZIP"situada al principi de l'enunciat del problema. Per exemple, al fitxer d'entrada `sample.inp` li correspon la sortida correcta `sample.cor` i podeu comparar la sortida del vostre programa `program.exe` amb la que conté `sample.cor` amb la comanda:

```
./program.exe < sample.inp | diff - sample.cor
```

4.1 Exemples d'ús de la STL

Llegiu-vos el document "Exemples d'ús de la STL" que trobareu com a entrada a l'apartat *Documentació addicional* dins de *Laboratori i pràctiques* del curs PRO1. Concretament, mireu-vos el primer apartat que explica com es fa la lectura de dades fent ús de `getline` i `istringstream`.

`getline`: Permet la lectura de dades línia a línia.

`istringstream`: Permet realitzar amb molta facilitat la lectura de paraules separades per espais o tabuladors.

A partir d'aquesta sessió podeu provar els exercicis sobre la plataforma Jutge. Accediu al curs *Programació 1 (EPSEVG)* i veureu una llista de problemes. Seleccioneu el problema a resoldre, l'implementeu i envieu la vostra solució al Jutge. Podeu enviar-la diverses vegades fins que la compilació i execució siguin correctes. El mateix Jutge us retornarà el veredicte:

```
AC      = Accepted
PE      = Presentation Errors
WA      = Wrong Answer
IC      = Invalid Character
EE      = Execution Error
CE      = Compilation Error
NC      = Noncompliant Solution
Pending = Pending Submission
SE      = Setter Error
SC      = Scored
IE      = Internal Error
FE      = Fatal Errors
```

Tingueu en compte els casos especials com estructures buides o amb un únic element. Els jocs de prova privats solen contenir aquests casos especials, per això és possible que la vostra implementació passi amb èxit els jocs de prova públics però no els privats.

Alguns exercicis de jutge ofereixen un o varis fitxers públics com, per exemple, el programa principal o el fitxer `Makefile` que conté les instruccions de compilació i muntatge. Els podeu descarregar cliclant la icona "Get public files"situada al principi de l'enunciat del problema.

Recordeu que per fer la majoria d'exercicis de la plataforma `Jutge` us pot ser útil la "lectura línia a línia" descrita al document "Exemples d'ús de la STL".

4.2 Estructures lineals de la STL

En aquesta sessió treballarem amb les estructures lineals que hem vist fins ara a teoria: piles, cues i llistes. Per aquestes estructures farem servir la Standard Template Library (STL) del C++ i les seves classes corresponents `stack`, `queue` i `list`, per això no haurem de mencionar els seus arxius a l'hora de muntar els nostres programes. Cal tenir en compte que en aquestes classes no es controla si les precondicions de les operacions se satisfan, per això ens interessarà usar l'opció de compilació `-D_GLIBCXX_DEBUG` per obtenir informació addicional en cas de `segmentation fault`, opció que ja inclou l'àlies `pro1++`.

Al directori que conté els arxius d'aquesta sessió hi trobareu els fitxers `stackIOint.hpp`, `queueIOint.hpp` i `listIOint.hpp` amb les operacions de lectura i escriptura per aquestes estructures (instanciades amb `int`). I als corresponents fitxers `...IOint.cpp` les seves implementacions. Noteu que aquestes operacions no poden ser genèriques, doncs depenen directament del tipus contingut dins les estructures.

4.3 Ús de la classe `stack`

A les classes de teoria hem vist exemples d'ús de la classe `stack` quan hem treballat amb piles d'enters. Ara volem fer exercicis similars sobre piles de punts o d'altres tipus i, per tant, caldrà implementar les operacions de lectura i escriptura per a piles instanciades amb la classe `Punt` o altres tipus.

4.3.1 Exercici del `Jutge`: Cerca en una pila de punts (X96304)

Heu d'implementar la cerca d'un punt en una pila de punts. Com a entrada hi haurà els punts que formen la pila i varis punts que cercarem. Com a sortida es mostrarà l'estructura de la pila de punts i, per cada punt a cercar, un missatge indicant si s'ha trobat o no.

Observació

A més de la solució en el fitxer `program.cpp`, heu d'implementar els fitxers `stackIOpunt.hpp` i `stackIOpunt.cpp` amb les operacions de lectura i escriptura per a piles instanciades amb `Punt`. Podeu agafar com a model els fitxers `stackIOint.hpp` i `stackIOint.cpp` que contenen les operacions de lectura i escriptura per a piles d'enters.

Heu d'enviar la solució comprimida en un fitxer `.tar`:

```
tar cvf program.tar program.cpp stackIOpunt.hpp stackIOpunt.cpp
```

Observeu que per compilar us donem el `Makefile` i el mòdul `Punt`.

4.3.2 Exercici del `Jutge`: Comprovant parèntesis (P69643)

Feu un programa que comprovi la correcta parentització d'unes quantes paraules usant una pila de caràcters. L'entrada consisteix en diverses paraules no buides formades amb els caràcters '(', ')', '[' i ']'. Per a cada paraula, cal escriure si la parentització és correcta o no.

Exemple:

```
Input           Output

()[]()          ()[]() es correcta
[(])            [(]) es incorrecta
[](             []( es incorrecta
((()))          ((()) es incorrecta
(([]()[()]))    (([]()[()])) es correcta
```

4.4 Ús de la classe `queue`

A les classes de teoria hi ha exemples d'ús de la classe `queue` instanciada amb enters. Ara volem fer exercicis similars sobre cues de punts o d'altres tipus.

4.4.1 Exercici del `Jutge`: Conversió d'una cua de punts en una pila (X44678)

Heu d'implementar la conversió d'una cua de punts en una pila de punts. El primer element de la cua ha de convertir-se en el cim de la pila i així successivament. Com a entrada hi haurà els punts que formen vàries cues. Com a sortida, per cada cua de l'entrada, es mostrarà l'estructura de la cua i l'estructura de la pila obtinguda al convertir la cua en pila.

Observació

A més de la solució en del fitxer `program.cpp`, heu d'implementar els fitxers `stackIOpunt.hpp`, `stackIOpunt.cpp`, `queueIOpunt.hpp` i `queueIOpunt.cpp`, amb les operacions de lectura i escriptura per a piles i cues instanciades amb `Punt`. Podeu agafar com a model els fitxers `stackIOint.hpp`, `stackIOint.cpp`, `queueIOint.hpp` i `queueIOint.cpp` que contenen les operacions de lectura i escriptura per a piles i cues d'enters.

Heu d'enviar la solució comprimida en un fitxer `.tar`:

```
tar cvf program.tar program.cpp stackIOpunt.hpp stackIOpunt.cpp
    queueIOpunt.hpp queueIOpunt.cpp
```

Observeu que per compilar us donem el `Makefile` i el mòdul `Punt`.

4.5 Ús de la classe `list`

A les classes de teoria hi ha exemples d'ús de la classe `list` instanciada amb enters. Ara volem fer exercicis similars sobre llistes de punts o d'altres tipus. Tingueu en compte l'especificació de `list` del fitxer `Especificacions_PCLA.pdf`.

4.5.1 Exercici del `Jutge`: Distribuir els elements d'una llista en dues llistes (X64830)

Escriviu una funció `separa` que, donada una llista *lp* de punts i un real *x*, torni dues llistes: *lp1* amb els punts de *lp* amb la coordenada x estrictament menor que *x*, i *lp2* amb els punts de *lp* amb la coordenada x estrictament major que *x*. *lp* ha de quedar buida. Observació: Si algun punt a la llista *lp* té la coordenada x igual a *x*, no apareixerà en cap de les llistes de sortida.

També heu de definir i implementar les operacions de lectura i escriptura de llistes de punts: `listIOpunt.hpp` i `listIOpunt.cpp`.

4.6 Exercicis complementaris

Us suggerim que feu pel vostre compte els següents exercicis per tal de millorar les vostres habilitats en programació i anar més ben preparats pels controls i activitats de l'assignatura.

4.6.1 Exercicis de cerca i ordenació del `Jutge`

Feu els tres problemes de l'apartat *Cerca i Ordenació* del curs *Programació 1 (EPSEVG)* (tots ells són algorismes que heu vist a l'assignatura prèvia *Fonaments de Programació*):

- *First occurrence* (P84219). Fixeu-vos que, atès que el vector d'entrada està ordenat, es pot fer de forma eficient fent una cerca dicotòmica: Mirant sempre l'element central de l'interval del sector a cercar, i decidir en funció del seu valor si cal cercar a l'esquerra o a la dreta d'aquest element central. Cal fer una variant de l'algorisme de cerca dicotòmica estàndard, ja que com que ens demanen la primera ocurrència del valor, en cas de que l'element central sigui igual al valor, encara no l'hem trobat i caldrà seguir mirant també a la seva esquerra.
- Ordenació per bombolla (P15549)
- Ordenació per inserció (P41412)

Si no teniu clar com funcionen els diferents algorismes d'ordenació us pot ser útil veure les animacions de `www.toptal.com/developers/sorting-algorithms` on es mostren els diferents algorismes treballant amb diferents conjunts de dades inicials.

4.6.2 Exercici del `Jutge`: Inversió separada amb piles (P13304)

Useu piles per invertir seqüències d'enters. Cal escriure primer els nombres parells en ordre invers, i després els nombres senars en ordre invers.

Per resoldre aquest exercici, els únics contenidors que hauríeu d'usar són piles.

Exemples:

Input	Output
`0 1 2 3 4 5 6 7`	`6 4 2 0 7 5 3 1`
`10 10 20 10`	`10 20 10 10`
`3 15 7 11`	`11 7 15 3`
`-1 -2 -3 -4`	`-4 -2 -3 -1`

4.6.3 Exercici del `Jutge`: Notació polonesa inversa (P52023)

Avalueu expressions aritmètiques donades en l'anomenada notació polonesa inversa, on primer apareixen els operands i després l'operador corresponent, fent innecessari l'ús dels parèntesis. Per exemple, l'expressió

```
3 + 154
```

en notació polonesa inversa es dona com

```
3 154 +
```

És a dir, primer es donen els dos operands, i després l'operador corresponent. Una expressió més complicada com ara

```
((3 + 4) * (2 - 8)) + (2 + 5)
```

es dona com

```
3 4 + 2 8 - * 2 5 + +
```

Exemples:

Input	Output
3 154 +	157
3 4 + 2 8 - * 2 5 + +	-35
99	99

4.6.4 Exercici del Jutge: Cues d'un supermercat (1) (P90861)

Simuleu el comportament de les cues d'un supermercat on hi ha n cues, cadascuna amb els seus clients. Els clients poden arribar i sortir de cadascuna de les cues.

4.6.5 Exercici del Jutge: Cerca en una llista de parells d'enters (X29544)

Donats una llista de parells (potser amb repeticions) de nombres enters (sempre acabada amb el parell *0 0*), i un número N, cal calcular el nombre d'aparicions d'N com a primer element dels parells de la llista i la suma dels seus companys. Si l'element no existeix, definim la suma dels companys com a zero.

Observeu que per als parells d'enters us donem la classe `ParInt` que detecta si el parell llegit és *0 0* i que per compilar us donem el `Makefile`.