VI. GELENEKSEL ODTÜ BİLGİSAYAR TOPLULUĞU ÜNİVERSİTE ÖGRENCİLERİ ARASI PROGRAMLAMA YARIŞMASI ÖN ELEME SORULARI

(Ayrıntılar için: http://www.cclub.metu.edu.tr/yarisma adresine bakınız)

1.Soru: Banka Şifresi

Kahramanımız Süpzek'in eşi ve çocukları BCU örgütünce kaçırılmıştır. Süpzek'den fidye olarak istedikleri ise ABC Bankası'nın şifreleme sistemini kırmasıdır. Süpzek uzun uğraşlar sonucunda bankanın şifreleme sisteminin, bankanın belirlediği bir grup asal sayıca belirlendiğini farkeder. Öyle ki, bir kişinin şifresi bu asal sayılardan istediğimiz kadarını istediğimiz adette çarparak elde edilen sayılardan en küçük k'ıncı sayıdır (k: müşteri numarası). Örneğin bankanın belirlediği asallar {3, 7, 13} olsun. Müşteri numarası 5 olan kişinin şifresi: {3, 7, 3*3, 13, 3*7, 3*3*3,....} kümesinin 5'inci elemanı olan 21'dir.

Sizden istenilen verilen asallarla müşteri numarası verilen kişinin şifresini bulan bir program yazmanızdır.

Varsayımlar

- Asalların sayısı: $3 \leq n \leq 100$.
- Müşteri numarası: $1 \leq k \leq 40$.

Girdi (banka.gir)

İlk satırda bankanın belirlediği asalların sayısı(n) verilecektir. İkinci satırda bu n asal, aralarında bir boşluk bırakılarak verilecektir. Üçüncü ve son satırda ise müşteri numarası(k) verilecektir.

Çıktı (banka.cik)

k nolu müşterinin şifresini vermelidir.

2.Soru: Birlikte Bir Otomata Yapalım

Bir 'düzenli ifade' a'dan z'ye kadar karakterleri, '*' işaretini ve parantezleri '(', ')' içerir. '*' işareti bir harfi veya bir harf grubunu düzenli bir şekilde tekrar ettirir. Mesela, 'ab(ca)*d' ifadesinde, '(ca)*', ifadede 'ca' n (0<=n<∞) defa tekrar edilebilir anlamına gelir. Yani bu ifade, 'abd', 'abcad', 'abcacad', 'abcacacad', ... ifadelerini kapsar demektir.

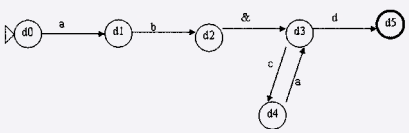
Amacımız düzenli ifadeyi tanıyan bir OBT otomatası yapmak. Bir OBT otomatası, düzenli bir ifadenin başından başlar ve bu ifadenin kapsadığı her kelimeyi sonuna kadar götürüp bitebilirse, düzenli ifade dışında da başka bir şey kapsamıyorsa, bu otomata düzenli ifadeyi tanıyor demektir. OBT otomatası 5 şeyden oluşur.

- **durumlar:** Düzenli ifadenin her hangi bir konumundayken içinde bulunulan otomata üzerindeki konuma durum denir. Otomatanın kapsadığı durumları belirtir.
- **alfabe:** Düzenli ifadenin içerebileceği tüm karakterlerdir.
- **başlangıç durumu:** Otomatanın başlatıldığı durum.
- **geçiş fonksiyonu:** Bir durumdan başka duruma hangi koşullarda geçileceğini belirtir.
- **bitiş durumu:** Otomatanın bittiği durum.

Örnek

- **durumlar:** d0,d1,d2,d3,d4,d5
- **alfabe:** a,b,c,d
- **başlangıç durumu:** d0
- **geçiş fonksiyonu:** (d0,a)=d1, (d1,b)=d2, (d2,&)=d3, (d3,c)=d4, (d4,a)=d3, (d3,d)=d5. '&' hiç harf kullanmadan başka bir duruma geçiş için kullanılır. Burada d2'den d3'e hiçbir harf kullanılmadan geçildi demektir.
- **bitiş durumu:** d5

Yukarıda belirtilen otomatanın çizimi;



Otomata Hazırlama Yöntemi

1. Alfabe, düzenli ifadenin kapsadığı harfler kümesi, belirlenir.
2. Başlangıç durumu üretilir.
3. Yıldızlı olmayan kısımlarda;

a. Bir harf için bir durum üretilir.

b. Eski durumdan yeni üretilen duruma bir harf ile gelinir.

Kısaca, eski durum d1, yeni durum d2 ve o harf de 'b' harfi ise geçiş fonksiyonu kümesine (d1,b)=d2 eklenir.

4. Yıldızlı kısımlar için;

a. Yıldızlı kısmın parantezi göründüğü anda yine yeni bir durum üretilir ama yeni duruma geçiş harfi boş (&) olmalıdır. Yukarıdaki örnekte d2 durumundan d3 durumuna '&' ile gidilmiştir.

b. Son harf hariç, her harf için yıldızsız kısımda yapılan işlem yapılır.

c. Son harfte, o anki durumdan yıldızlı ifadenin başındaki duruma gidilir. Yukarıdaki örnekte (ca)* yıldızlı kısımdır. Yıldızlı ifadenin başlangıç durumu d3'dür. Yıldızlı kısımdaki son harf 'a' olduğundan d4 durumundan, d3'e 'a' ile donülmüştür.

5. En son gelinen durum bitiş durumudur.

Varsayımlar

- Düzenli ifadenin boyu 50 karakteri geçmeyecektir.
- İfadedeki yıldızlı kısımlar parantez içinde olacak, yıldızlı ifadeler dışında parantez başka yerde kullanılmayacak.
- Toplam durum sayısını d kabul edersek, durum numaraları 0'dan d-1'e kadardır.

Girdi (otomata.gir)

Tek satırdan oluşmaktadır. Sadece düzenli ifadeyi içerir.

Çıktı (otomata.cik)

İlk satırda durum sayısı (d) ve geçiş fonksiyon kümesinin eleman sayısı (f) verilmelidir. İkinci satırda aralarında boşluk olmaksızın alfabe karakterleri

verilmelidir. Üçüncü satırda başlangıç ve bitiş durum numaraları olmalıdır. Ondan sonraki f satırda, geçiş fonksiyon kümesinin elemanları şu şekilde verilmelidir;

eski_durum_numarası harf yeni_durum_numarası

3.Soru: Robot

OBT simülasyon robotu şu şekilde çalışmaktadır: Robota bir grup yapılması gereken iş listesi verilmektedir. İşler baska işlere bağlı olabilir ve bu durumda diğerleri yapılmadan o iş yapılamaz. Mesela A işi B işine bağlı ise B yapılmadan A yapılamaz. OBT robotunda bağlı işler listesi vardır ve robotun aynı anda yapabileceği iş sayısında bir sınır yoktur. Sizden istenen robotumuzun işleri hangi sıra ile yaptığını bulmanız.

Girdi

Girdi robot.gir isimli bir metin dosyasıdır. Bu dosyanın ilk satırında, kaç tane işin bulunduğunu gösteren bir pozitif tamsayı (N) ve kaç adet bağlılık ilişkisi olduğunu gösteren bir pozitif tamsayı (E) vardır. Bunu izleyen E satırın herbiri, bir boşlukla ayrılmış iki pozitif tamsayı (sırasıyla, P ve Q) taşır. Bunun anlamı P numaralı iş yapılmadan Q numaralı iş yapılamaz. Bundan sonra boş bir satır bulunur. Daha sonraki satırda da robotun yapması gereken kaç adet iş olduğunu gösteren bir pozitif tamsayı (K) bulunur. Dosyanın son satırında ise bu işler, tekrarlama olmaksızın, bulunur.

Çıktı

Çıktı robot.cik isimli bir metin dosyası olmalıdır. Dosyanın ilk satırında işlerin kaç birim zamanda yapıldığını belirten bir pozitif tamsayı (G) olmalıdır. Bunu izleyen G adet satır, sırasıyla her zaman diliminde yapılan işleri içerir. Her satırdaki ilk sayı (L) o zaman diliminde kaç tane iş yapıldığını göstermektedir. Ondan sonra gelen L tane sayı da yapılan işleri belirtmektedir. Bir zaman diliminde yapılan işlerin numaraları birer boşlukla ayrılmış olarak gösterilmelidir. Bir zaman diliminde yapılan işlerin gösterilme sırası önemli değildir.

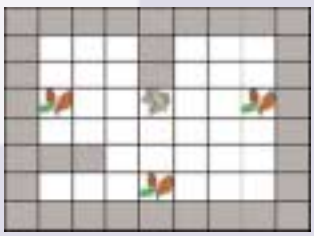
Varsayımlar

- 0 < N <= 1000
- 0 < K <= N
- Her bir iş bir birim zamanda yapılmaktadır.

Özel Soru: Kurbağa Operasyonu

Kurbağa Süpzek bir labirentte yaşamaktadır. Labirentte engeller ve otlar bulunmaktadır. Süpzek'in amacı labirentteki tüm otları bitirmektir.

Yapabileceği 4 hamle vardır; sagadon, soladon, zipla ve otye. 'sagadon' hamlesi ile 90 derece sağa, 'soladon' hamlesiyle 90 derece sola döner. 'zipla' hamlesi Süpzek'in bulunduğu yönde 1 kare ilerlemesini sağlar. 'otye' ile Süpzek bulunduğu karedeki otu yer.



Örnek

Yukarıdaki labirentte Süpzek'in başlangıçtaki yönünün yukarı doğru olduğunu varsayarsak, bir çözüm şu şekildedir;

```
soladon
zipla
zipla
zipla
otye
soladon
soladon
zipla
zipla
zipla
zipla
zipla
zipla
zipla
otye
sagadon
zipla
zipla
zipla
sagadon
zipla
zipla
zipla
otye
```

Sizin programınız buna benzer bir çıktı üretmelidir. Bu komutlar ZBasüx dili komutlarıdır. ZBasüx dili çok gelişmiş olduğu için fonksiyon kullanımına da sahiptir. Fonksiyon şablonu aşağıdaki şekildedir;

fonksiyon fonksiyon_ismi
komut
komut
...
...
son

Fonksiyon çağırma işlemi de 'cagir fonksiyon_ismi' şeklinde yapılır.

*** Sorunun amacı mümkün olduğu kadar az satırdan oluşan çıktı dosyası üretmektir.**

Varsayımlar

- Labirent boyutları 7<=n<=30, 7<=m<=30'dir.
- Süpzek'in başlangıçtaki yönü yukarıdır.

Kısıtlar

- Bir tane ana fonksiyon olmalıdır ve adı 'ana' dır. Programınız C dilindeki gibi ana fonksiyondan işletilmeye başlanır.
- Fonksiyonlar sadece ana fonksiyondan çağırılabilir, fonksiyon içinden fonksiyon çağırımı yapılamaz. Ana fonksiyon kendini de çağıramaz.
- Ana fonksiyon dışındaki fonksiyon isimleri, 'sagadon', 'soladon', 'zipla', 'otye', 'fonksiyon', 'cagir', 'son', 'ana' olamaz.
- Fonksiyon isimlerinin maksimum uzunluğu 5 karakter olabilir.
- Ana fonksiyon her zaman en son fonksiyon olmalıdır.
- Her satır sadece bir komuttan oluşabilir. Komutlar: fonksiyon, son, cagir, sagadon, soladon, zipla ve otye'dir.
- Çıktı dosyası 5000 satırdan kısa olmalıdır.
- Süpzek duvara zıplayamaz ve labirent dışına da çıkamaz.

Girdi (kurbaga.gir)

İlk satırda labirentin boyutları (n,m) verilecektir. Ondan sonra gelen n satırda her biri m uzunluğunda olan karakter dizisi şeklinde labirent verilecektir. Girdi dosyasında;

X: duvar,
K: kurbağa Süpzek,
O: ot,
-: boş alanı belirtmektedir.

Çıktı (kurbaga.cik)

Süpzek'in tüm otları yemesini sağlayacak, mümkün olduğunca az sayıda satırdan oluşan komut dizilimini içermelidir. Çıktı dosyasında her satırda bir komut olmalıdır, başlarında ve sonlarında boşluk olmamalıdır.

Değerlendirme

Programlar iki kategoride değerlendirilecektir; üretilen en kısa kod ve en hızlı çözüm.