ОДСЕК ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО
АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА 2

II ДОМАЋИ ЗАДАТАК 2011-2012

Напомена: студенти самостално раде домаћи задатак. Од студената се **очекује** да по потреби консултују додатну литературу. На одбрани задатка, студенти треба да покажу изворни код програма који се успешно преводи. На одбрани се од студената могу тражити модификације показаног програма као и детаљно објашњење начина рада имплементираних алгоритама.

1. [35 поена] Оптимизација система претраге

Једна од услуга која је бесплатно доступна као основни сервис Интернета јесте могућност да се установи да ли је одређен сајт доступан (не постоји регистровани корисник који је његов власник), односно да се установи ко је власник одређеног сајта.

Више Интернет компанија се бави унапређивањем поменутог сервиса у циљу поједностављења достављања тражених информација у вези сајтова крајњим корисницима. *IZS*, једна од тих компанија, ангажовала Вас је на решавању техничког проблема који је настао огромним повећањем броја корисника Интернета. Наиме, у почетку је број корисника Интернета био релативно мали, и *IZS* се задовољила формирањем једноставне уланчане листе за смештање података о сајтовима и власницима. Како је број корисника порастао, време потребно за претрагу је такође расло, а квалитет услуге коју пружа компанија је самим тим опао, што је даље довело до пада угледа компаније.

Ваш задатак јесте да реализујете ефикасан систем претраге који нуди следеће могућности:

- тражење власника датог сајта (регистрован власник не мора да постоји, такав сајт се сматра доступним)
- тражење свих сајтова који су у власништву датог корисника
- додавање новог власника датог сајта, уз задавање датума истека најма сајта (операција могућа само ако сајт нема власника)
- брисање евиденције о власништву над датим сајтом
- проналажење свих сајтова чији најам истиче у року од 7 дана од задатог датума
- брисање свих власника сајтова чији је најам истекао пре задатог датума

Систем треба да обезбеди да време претраге (било успешне или неуспешне) слабо зависи од броја регистрованих сајтова или корисника. Сматрати да један власник може имати више сајтова у свом власништву, али да један сајт може имати само једног власника. Сматрати да се назив сајта састоји од произвољне комбинације алфанумеричких латиничних карактера и специјалних знакова: повлака, тачка и плус. Сматрати да се име корисника састоји искључиво од латиничних карактера и специјалних знакова: тачка и размак.

2. [45 поена] Адаптивна хеш-табела

Написати класу у језику C++ која имплементира хеш табелу за смештање кључева произвољног типа, која кориснику пружа следеће операције:

- стварање празне хеш табеле
- уметање кључа и одговарајућег информационог садржаја у табелу (враћа логички индикатор успеха; пријављује неуспех ако кључ већ постоји)
- претрагу на задати кључ у табели (враћа адресу информационог садржаја, односно NULL у случају да кључ не постоји)
- брисање кључа из табеле

С обзиром на то да су кључеви, као и пропратни информациони садржај, произвољног типа, неопходно је класу реализовати применом шаблона. Једноставан начин да се униформно подржи било који тип кључа јесте да се за сваки тип кључа обезбеди функција `int hash(tip)` (пријатељска, у случају класног типа), која враћа целобројну вредност која се потом тумачи као стварна вредност кључа. Као типове кључева, од простих типова подржати: `int`, `float`, `double` и `char *`.

Напомена: `hash()` функције за целобројни тип и за реални тип једноструке прецизности би могле да буду реализоване на следећи начин (подразумевана дужина целобројног података је 32 бита):

```
int hash(int i) { return i; }
int hash(float f) {
   int *t = (int *)&f; return *t & 0x7FFFFFFF;
}
```

Исправна реализација би морала да у обзир узме величину целобројног типа на конкретном рачунару.

Табела аутоматски треба да прати своје перформансе и да се прилагођава подацима тако да обезбеди ефикасно извршење операције претраге (тако да траје што краће). То ће се постићи тако што ће се, уколико дође до губитка перформанси, сви кључеви преместити у нову табелу која ће имати другачије карактеристике (величину) погоднije за конкретни скуп кључева уметнут у табелу. При том треба водити рачуна и о ефикасној употреби меморије.

Конкретно, као индикаторе перформанси треба користити:

- попуњеност табеле (табела се аутоматски проширује када попуњеност пређе задату границу)
- просечан број приступа приликом успешне или неуспешне претраге (табела се проширује у случају да је број приступа већи од вредности која се израчунава на основу броја уметнутих кључева).

За успешно решење овог дела задатка, потребно је конкретно формулисати и имплементирати наведена два критеријума за аутоматско проширење величине табеле.

3. [20 поена] Сортирање алгоритмом "мађионичара"

Млади Пера је још као дечак открио да поседује видовњачке способности. Покушавао је да их усаврши, али то је успео само до неке мере. Сада је одлучио да се опроба као мађионичар и смислио један занимљив трик, који је описан у наставку.

Некоме из публике Пера даје шпил карата. Та особа измеша шпил и врати Пери. Онда Пера, без гледања у карте, поново меша и премешта карте у датом шпилу тако да се након тога добије уређен шпил, где су карте поређање од најмање до највеће (ради једноставности узето је да су вредности карата од 0 до 51). Прецизније, Пера може да изврши само операције пресецања шпила на произвољном месту (када два дела шпила настала пресецањем замене места, без промене поретка карата унутар делова) и операције замене места две произвољне карте из шпила (остале карте не мењају позицију). Идеално, Пера би својим видовњачким способностима успео да сазна редослед карата у шпилу добијеном од особе из публике и онда директним премештањем карата успео да изведе трик. Међутим, како је претходно речено, Пера није успео до краја да развије своје способности и може да сазна само редослед и вредност првих 6 карата са врха шпила и првих 6 карата са дна шпила.

Помозите Пери да успешно изведе овај трик, уз што је могуће мањи број премештања карата. Треба написати програм који за дати (измешани) шпил карата, као резултат, даје редослед операција премештања карата којим се добија сортиран шпил.

Решење проблема се састоји у реализовању функције `sort(Spil s)` која прима (измешан) шпил карата `s` и сортира га (не прави сортирану копију, већ сортира шпил прослеђен као аргумент). Ова функција треба да користи помоћне структуре података да би пратила позицију сваке познате карте у шпилу. Приликом решавања проблема, сматрати да не постоји ограничење у количини меморије или времену потребном да се изврши сортирање. Једини параметар успешности јесте број операција са картама (пресеци и замени). Мањи број операција је успешнији. Класа `Spil` представља шпил карата који се састоји од 52 карте. Декларација метода класе `Spil` је дата у наставку. Као део решења, најпре је потребно потпуно реализовати класу `Spil`.

```
class Spil{
public:
      Spil();//napravi izmesan spil karata
      void zameni(int a, int b);//zamena mesta karata na pozicijama a i b
      void preseci(int i); // preseca spil na mestu i
      const int[] vrh();//vraca vrednosti 6 gornjih karata spila
      const int[] dno();//vraca vrednosti 6 donjih karata spila
};
```

Напомена: Из поставке се може закључити да вредност карте (0 до 51) одређује позицију карте у сортираном шпилу. Међутим, распоред карата није познат и главни проблем јесте одгонетнути га, уз ограничења из поставке задатка, у што је могуће мањем броју корака. Пресецањем шпила, карте непосредно до места пресека постају познате, зато што се премештају на врх, односно на дно шпила.

Једно од једноставнијих (и мање ефикасних) решења би било да се шпил више пута "сече" на различитим местима, и након сваког пресецања се бележе карате са врха и дна шпила. Поступак би се поновио потребан број пута док се не сазна распоред свих карата у шпилу. Након тога је потребно преместити карте на њихове праве позиције.

У наставку је дат пример са редукованим шпилом од 11 карата и могућношћу "вићења" до три карте са врха и дна шпила. Карте означене **подебљаним** словима су познате. Карте означене црвеном бојом се налазе на својој коначној позицији.

Иницијални распоред карата	preseci(5)	zameni(0,2)	zameni(1,3)	zameni(8,10)	zameni(8,9)
6	**2**	**0**	**0**	**0**	**0**
7	**3**	**3**	**1**	**1**	**1**
10	**0**	**2**	**2**	**2**	**2**
8	**1**	**1**	**3**	**3**	**3**
9	**4**	**4**	**4**	**4**	**4**
2	**5**	**5**	**5**	**5**	**5**
3	**6**	**6**	**6**	**6**	**6**
0	**7**	**7**	**7**	**7**	**7**
1	**10**	**10**	**10**	**9**	**8**
4	**8**	**8**	**8**	**8**	**9**
5	**9**	**9**	**9**	**10**	**10**

Очекивани резултат програма:

```
preseci(5)
zameni(0,2)
zameni(1,3)
zameni(8,10)
zameni(8,9)

Presecanja: 1
Zamena: 4
```