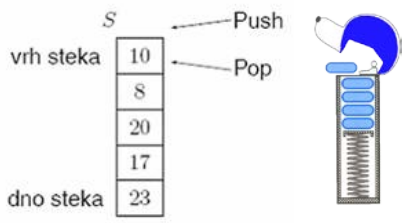
STEK



STEK - kolekcija podataka sa kojima se radi po LIFO(Last In First Out) principu;

Procedure za rad sa stekom

Da li je stek prazan
Postavi element na vrh steka (tzv. operacija PUSH)
Skini element sa vrha steka (tzv. operacija POP)

Stek je prazan ako mu se vrh nije pomerio iz inicijalnog stanja.

Funkcijom push se formira sadrzaj steka - dodavanjem elemenata na vrh steka.
Funkcijom pop elementi se uzimaju sa steka po LIFO redosledu (najpre se uzima element sa vrha steka)
Upotreba steka:
IV glava K&R - primer sa kalkulatorom i izračunavanje u postfiks notaciji (podsećanje:
AB+ ~ A+B, ABCD-*+EF-G*H/+ ~ A+B*(C-D)+(E-F)*G/H) ,
provera uparenosti HTML etiketa,
provera uparivanja zagrada (leksička analiza,)
kod editora teksta (editovanje reda),
pri rekurziji i pozivu potprograma

Stek ograničenog kapaciteta (sa **MAX** elemenata) se može implementirati pomoću niza **Stek[0..MAX-1]** tako da element Stek[0] je na dnu steka, element Stek[MAX-1] je na vrhu steka. Ako se pokuša umetanje elementa na poziciju MAX, nastaje prekoračenje (overflow).

STL
Da bi mogao da se koristi stek kao struktura podataka u STL-u, onda se na počeku programa mora uključiti direktiva #include <stack>

Stek sadrži elementi odredjenog tipa. Ako želimo da deklarišemo stek koji će sadržati stringove kao podatke, deklaraciju pišemo:
stack<string> s;

Naravno, kad god se koristi string kao tip podataka, mora se ukljuciti direktiva
#include <string>

Ako želimo da deklarišemo stek koji ce sadržati cele brojeve kao podatke, deklaraciju pišemo: stack<int> s;
ili uopšte za proizvoljni tip podataka (primitivni tip podataka, klasa) T
stack<T> s;

Dozvoljeni metodi

- Za proveru da li je stek prazan koristi se metodata empty, tj.

s.empty() //->bool

Ova metoda vraća true ako je stek prazan ili false uslučaju da stek ima barem jedan element.

- Za proveru koliko ukupno elemenata se nalazi u steku se koristi metoda size

s.size() //->int

- Za unos novog elementa t tipa T na vrh steka se koristi metoda push, na primer

s.push(t) //->void

- Za čitanje i skidanje elementa sa vrh steka se koristi metoda pop, na primer

s.pop() //->void

- Ako želimo samo da pročitamo element na vrhu steka, ali ne i da ga odstranimo sa vrha, onda se koristi metod

s.top() //->T

```
//direktiva za ukljucivanje potrebnih klasa
#include <iostream>
#include <stack>
#include <string>
using namespace std;
int main() {
//deklarisemo stek strukturu sa elementima tipa string
stack<string> s;
//Unesimo tri stringa i sacuvajmo u steku
s.push("NISKA 1");
s.push("NISKA 2");
s.push("Niska 3");
string w;
cin>>w;
s.push(w);
//Stampajmo broj clanova steka koji smo formirali
cout << "Broj                = " << s.size() << endl;
//Pitajmo da li stek nije prazen
while (!s.empty()) {
//Stampanje sadrzaja sa vrha steka
cout << s.top() << endl;
//Iscitavanje vrha steka i skidanje elementa sa vrha steka
s.pop();
}
return 0;
}
```

Zadaci

1. Sadržaj ulazne datoteke (stdin) je aritmetički izraz koji može sadržati zagrade {, [i (. Napisati program koji učitava sadržaj ulazne datoteke i korišćenjem steka (engl. stack) utvrđuje da li su zagrade u aritmetičkom izrazu dobro uparene. Program štampa odgovarajuću poruku na standardni izlaz.

Ulaz

{[23 + 5344] * (24 - 234)} – 23

Izlaz

Zagrade su ispravno uparene.

Ulaz
{[23 + 5] * (9 * 2)} - {23}
Izlaz
Zagrade su ispravno uparene.

Ulaz
{[2 + 54) / (24 * 87)} + (234 + 23)
Izlaz
Zagrade nisu ispravno uparene.

Ulaz
{(2 - 14) / (23 + 11)}} * (2 + 13)
Izlaz
Zagrade nisu ispravno uparene.

Rešenje
1. Citamo znak po znak teksta do kraja ulaza (EOF).
2. Svaku otvorenu zagradu stavljamo na stek.
3. Kada naiđemo na zatvorenu zagradu, skidamo odgovarajuću otvorenu (poslednju stavljenu) sa steka.
Ako zatvorena zagrada ne odgovara otvorenoj, onda postoji greška u ugnježdavanju zagrada.
4. Ako je na kraju programa stek prazan i mi smo učitali sve znake teksta, onda su zagrade dobro uparene.
Skicirajmo stek za gore navedene test primere.

```
#include <iostream>
#include <stack>
using namespace std;
stack<char> s;
int main()
{
   char c;
   while(cin.get(c))
   {
      /* Ako je ucitana otvorena zagrada, stavlja se na stek */
   if (c == '(' || c == '{' || c == '[')
         s.push(c);
   /* Ako je ucitana zatvorena zagrada, proverava se da li je stek
     prazan i ako nije, da li se na vrhu steka nalazi odgovarajuca
     otvorena zagrada */
   else {
    if (c == ')' || c == '}' || c == ']') {
     if (!s.empty()  && ((s.top() == '(' && c == ')')
                      || (s.top() == '{' && c == '}')
                      || (s.top() == '[' && c == ']'))) {
       /* Sa vrha steka se uklanja otvorena zagrada */
       s.pop();
     } else {
```

```
        /* Dakle, zagrade u izrazu nisu ispravno uparene */
        break;
      }
    }
  }
 }

/* Ako je stek prazan i procitana je cela datoteka, zagrade su
    ispravno uparene. */
 if (s.empty()  && cin.fail())
   cout << "Zagrade su ispravno uparene.\n";
 else
   /* U suprotnom se zakljucuje da zagrade nisu ispravno uparene. */
   cout <<"Zagrade nisu ispravno uparene.\n";
   return 0;
}
```

2. **Napisati program koji će proveriti uparenost etiketa u ulaznoj HTML datoteci. Složena etiketa A (poseduje otvaracA i zatvaracA) je korektno ugnježdena u etiketu B ako je otvaracA iza otvaracB i zatvaracA ispred zatvaracB.Pretpostaviti da u ulaznoj datoteci nema prostih etiketa (onih koje nemaju zatvarac).Jednostavnosti radi, pretpostaviti da maksimalna dubina ugnježdenosti je do nivoa 3(dakle, nivo tipa <B><I>...</I> <I> </i> se pojavljuje, dok nivo tipa <B><I>... <a href="nesto.htm">...</A> </i> se ne pojavljuje). Pretpostaviti da nazivi etiketa su jednoslovni (P,Q,B,I,U,S,A,...).Pri nailasku na nekorektno uparenu etiketu, prekinuti dalju proveru i ispisati poruku o grešci na standardni izlaz.**

TEST PRIMER:

```
<P align="center">Pasus 1 </P>
<p><B><q>Citat 1</q> <A href="link1.htm"> Hiperveza 1 </a> <Q> Citat 2 </q> ... </b>...</p>
<p><q>Citat 2</Q> <A href="link2.html"> Hiperveza 2 </a> <B> <A href="link3.htm"> Hiperveza 3
</a>...</B> </P>
```

IDEJA: **Svaka otvorena etiketa stavi se na stek**

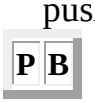
Pri nailasku na zatvorenu etikete proveravamo da li je stek prazan.
Ako jeste, nekorektna uparenost.
Takođe se proveri i da li se na vrhu steka za tekucu zatvorenu etiketu nalazi odgovarajuca otvorena etiketa.
Ako ne, nekorektna uparenost. Ako da, skine se otvorena etiketa sa vrha steka.

Prazan stek

push(P)

pop(), provera == pop()

push(P)



push(B)

P B Q

push(Q)

P B

provera </q > == pop()

```
#include <iostream>
#include <stack>
#include <cctype>
using namespace std;
stack<char> s;
bool greska=false; /*indikator nailaska na pojavu nekorektne ugnjezdenosti*/
int main()
{
   char c; /*znak sa ulaza*/
   char rez; /*rezultat skidanja sa steka*/
  while ( cin.get(c)  && !greska)
    if (c=='<')
    {   cin.get(c);
        if (isalnum(c) )  s.push(toupper(c));
         if (c=='/')  /*obrada zatvaraca, ali samo ako je oblika </slovo...> */
            {    cin.get(c);
               if (isalnum(c) && ! s.empty())
               {
                  rez=s.top(); s.pop();
                if (toupper(c) != rez)  {cout << "Greska: etiketa " <<  c << endl;
                                greska=true; }
               }
               else if (s.empty())  {cout << "Greska: lose ugnjezdene etikete " <<  endl;
                                greska=true; }
            }
    }

   if (s.empty() && !greska) cout << "\nKorektna uparenost unutar HTML datoteke\n";
   else  cout << "\nNekorektna uparenost unutar HTML datoteke\n";
   return 0;
}
```

3. maxStack

vreme	memorija	ulaz	izlaz
1 s	1000 Mb	standardni ulaz	standardni izlaz

Vaš zadatak je da implementirate standardne funkcije stack-a, tj. Push(), Seek() i Pop(), međutim potrebno je implementirati i funkciju Max() koja vraća najveći element na stack-u.

Ulaz
U prvom redu ulaza se nalazi broj Q, koji predstavlja broj upita koje je potrebno izvršiti. U narednih Q linija se nalaze upiti. Postoji 4 tipa upita:

- '1 x', gde je x broj koji se ubacuje na stack
- '2', izbacuje se element sa stack-a
- '3', potrebno je ispisati vrednost poslednjeg elementa na stack-u
- '4', potrebno je ispisati najveću vrednost koja se trenutno nalazi na stack-u

Izlaz
Na svaki upit tipa '3' i '4', potrebno je ispisati jedan broj koji predstavlja odgovor na upit.

Ograničenja
1 <= Q <= 1000000

n	$O(n^2)$	vreme
1	1	trenutak
100	10.000	trenutak
10.000	100.000.000	~1s
1.000.000	1.000.000.000.000	~3h

n	O(nlogn)	vreme
1	1	trenutak
100	700	trenutak
10.000	130.000	~0.005s
1.000.000	20.000.000	~0.5s

Primer

Ulaz	Izlaz
7 1 6 1 1 3 4 1 7 4 3	1 6 7 7

```
#include <iostream>
#include <stack>
using namespace std;
stack<int> stck, maxst;
int main() {
    int N, i, x, y, mx;
   // stck = stack<int>();
   // maxst = stack<int>();
    cin >> N;
    for(i=0; i<N; i++) {
        cin >> x;
        switch(x) {
```

```
            case 1:  //UBACUJEMO element na stek
                cin >> y;  stck.push(y);
                if(maxst.size()) { //ako maxst ima clanove
                    mx = maxst.top();  //mx je element na vrhu max steka
                    if (y > mx) {
                        maxst.push(y);
                    }
                } else {  // ako nema elemenata na vrhu max steka, onda postavi y
                    maxst.push(y);
                }
                break;
            case 2:     //SKIDAMO element sa steka
                y = stck.top();
                mx = maxst.top();
                if(y == mx) {    //ako element koji skidamo na steku je max, skidamo ga i sa max
steka
                    maxst.pop();
                }
                stck.pop();
                break;
            case 3:    //ispis vrha steka
                printf("%d\n", stck.top());
                break;
            case 4: //ispis vrha mAXIMALNOG steka
                printf("%d\n", maxst.top());
                break;
        }
    }

    return 0;
}
```

4. Poruka (word wrap like problem)

vreme	Memorija	ulaz	Izlaz
2 s	1000 Mb	standardni ulaz	standardni izlaz

Potrebno je implentirati sistem za kucanje poruka. Cela poruka se nalazi u jednoj liniji, i sačinjena je od velikih i malih slova engleske abecede.

Na početku vam je dat tekst, kao i pozicija kursora u tekstu. Postoje 4 tipa komandi koje se mogu proslediti vašem sistemu:

- '1' - pomeri kursor za jednu poziciju ulevo, ukoliko se kursor nalazi na početku teksta onda ne treba ništa da se uradi

- '2' - pomeri kursor za jednu poziciju udesno, ukoliko se kursor nalazi na kraju teksta onda ne treba ništa da se uradi
- '3 x' - ubaci karakter x na poziciju gde se nalazi kursor i posle izvršavanja upita, kursor treba da se nalazi iza ubačenog karaktera
- '4' - obrisi karakter iza kojeg se nalazi kursor, ukoliko se kursor nalazi na početku teksta onda ne treba ništa da se uradi (obrisati karakter ispred kursora)

Potrebno je izvršiti Q komandi i ispisati kako izgleda tekst nakon izvršenih Q komandi.

Ulaz
U prvom redu ulaza se učitava početna poruka.

U drugom redu ulaza se nalaze 2 broja, prvi označava poziciju karaktera u početnoj poruci iza kojeg se nalazi kursor (ukoliko je broj 0, onda se nalazi ispred prvog karaktera u poruci), a drugi broj je broj Q, tj. broj upita.

U narednih Q redova se nalaze komande u formatu opisanom u tekstu zadatka.

Izlaz
U prvi i jedini red izlaza potrebno je ispisati kako izgleda poruka nakon primenjenih Q komandi.

Ograničenja
dužina početne poruke <= 100000
0 <= Q <= 1000000
abcd -> |bcd->c|bcd-> cb|cd ->cbd|cd
Primer
Ulaz
abcd
1 4
4
3 c
2
3 d
IZLAZ
cbdcd

Hint😐

#include <cstdio>

```
#include <list>
using namespace std;
list<char> msg;
int main() {
    char c;
    while (true) {
        scanf("%c", &c);
        if (c == '\n')  break;
        msg.push_back(c);
    }

    int p, q;
    scanf("%d %d", &p, &q);
    list<char>::iterator it = msg.begin();
```

/* ucitano p označava poziciju karaktera u početnoj poruci iza kojeg se nalazi kursor*/

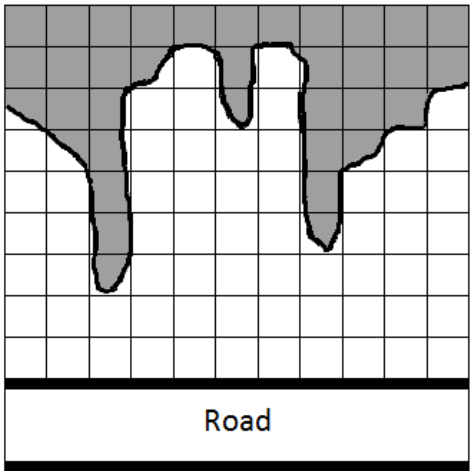
```
    for (int i = 0; i < p; i++) it++;

    int opt1;
    char opt2;

    for (int i = 0; i < q; i++) {
        scanf("%d", &opt1);
        switch (opt1) {
            case 1:
                if (it != msg.begin())// ukoliko se kursor ne nalazi na početku teksta
                    it--; // pomeri kursor za jednu poziciju ulevo
               break;
            case 2:
                if (it != msg.end()) //ako se kursor ne nalazi na kraju teksta
                    it++; // pomeri kursor za jednu poziciju udesno
                break;
            case 3:  //ubaci karakter opt2 i pomeri kursor
                scanf(" %c", &opt2);
                msg.insert(it, opt2);
                break;
            case 4:
                if (it != msg.begin()) {//ako kursor nije na pocetku teksta
                    it--;
                    it = msg.erase(it);  //obrisati karakter ispred kursora
                }
                break;
        }

    }

    for (list<char>::iterator jt = msg.begin(); jt != msg.end(); jt++)
            printf("%c", *jt);
        puts("");

    return 0;
}
```

RESENJE sa 2 steka (stek a sa karakterima ispred prve pozicije pos i stek b sa obrnutim redosledom karaktera iza prve pozicije pos)

```
#include <cstdio>
#include <stack>
#include <cstring>
#define N 1100100
#define M 10
using namespace std;

char in[N], out[N], x[M], c;

int main() {
        stack<char> a, b;
        int pos,q,l,i,j;
        scanf("%s", in);       l = strlen(in);
        scanf("%d %d", &pos, &q);

        for(int i=0; i<pos; i++) {   //stek a
                a.push(in[i]);
        }

        for(int i=l-1; i>=pos; i--) { //stek b
                b.push(in[i]);
```

```
	}


	for(int i=0; i<q; i++) {
		scanf("%s", x);
		switch(x[0]) {
			case '1':
				if(!a.empty()) {
					c = a.top();
					a.pop();
					b.push(c);
				}
				break;
			case '2':
				if(!b.empty()) {
					c = b.top();
					b.pop();
					a.push(c);
				}
				break;
			case '3':
				scanf("%s", x);
				a.push(x[0]);
				l++;
				break;
			case '4':
				if(!a.empty()) {
					//brise se  a.top()
					a.pop();
					l--;
				}
				break;
		}
	}

	j=i=a.size()-1;
	//dopuna izlazne poruke out stekom a i stekom b
	 while(!a.empty()) {
		c = a.top();
		out[i--] = c;
		a.pop();
	}
	out[l] = '\0';
	j++;
	while(!b.empty()) {
		c = b.top();
		out[j++] = c;
		b.pop();
	}

	printf("%s\n", out);

	return 0;
}
```

5. Добро је познато да је веома исплативо поседовати хотел близу морске обале. Зато је фирма International Ocean Investment закупила парче земље на обали Црног мора (налик на датој слици) и жели да изгради хотел –што је могуће већи. Из различитих разлога, основа хотела мора имати правоугаони облик. Зато фирма тражи неког да нађе правоугаоник највеће површине који може да се исцрта на парчету земље. У ту сврху, читав терен је подељен у N колона, а свака колона садржи једнаке квадрате (бели делови слике). Колоне су нумерисане узастопним бројевима 1, 2,…,N, гледано слева на десно, а правоугаоник треба да буде састављен од целих бројева таквих квадрата. Потом, за сваку колону је нађен број целих (белих) квадрата. Напишите програм **`maxarrea`**, који ће наћи површину највећег правоугаоника, састављеног од целих квадрата, који може да се постави на терену.



Улаз

Први ред **стандардног улаза** садржи цео број N ($N \leq 1\ 000\ 000$). У следећем реду је дато N целих бројева D_1, D_2, …, D_N – где D_I је број квадрата у колони I, $0 < D_I \leq 15\ 000$.

Излаз

Програм треба да испише на **стандардни излаз** нађену максималну површину.

Пример

Улаз

```
11
6 5 2 7 8 6 8 3 5 6 7
```

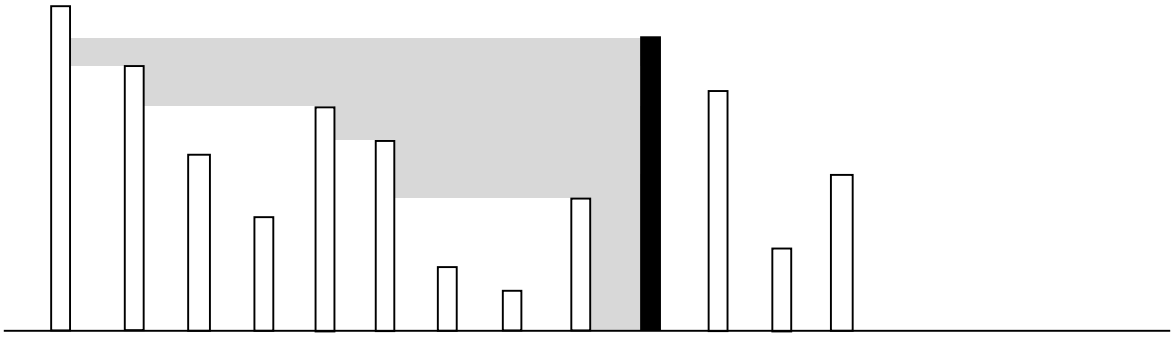
Излаз

```
24
```

6. Grad *X* se sastoji od *N* zgrada, poređanih u jedan red od zapada ka istoku. Zgrade su numerisane od 1 do *N*. **Svaka zgrada ima različitu visinu – ceo broj, redom $h_1, h_2, \ldots, h_N$.** Gradska uprava planira da sagradi kulu, koja će biti u istom redu sa zgradama (može da bude pre prve zgrade, između bilo koje dve zgrade ili nakon poslednje zgrade). Kula će izdavati saopštenja građanima. **Visina kule mora biti različita od visina svake zgrade i označena je celim pozitivnim brojem *H*.**

Zbog čudnih inženjerskih ideja, kula će emitovati signale samo ka zapadu tj. na početku reda zgrada. I signali su čudni – predstavljaju ih zraci koji se kreću horizontalno (paralelno površini tla, za koji smatramo da predstavlja pravu liniju) i oni se emituju duž cele kule(od vrha do podnožja). Otuda, možemo da zamislimo da kula izluči neprekidnu lentu signala čija širina je jednaka visini kule. Kada zrak udari zgradu, zaustavlja se. **Svaka zgrada prima signale koristeći prijemnik smešten na njenom vrhu.** Zgrada prima saopštenja ako najmanje jedan zrak dostigne njen prijemnik.

Drugim rečima, zgrada numerisana sa *i* će primiti saopštenje sa kule samo ako: zgrada *i* se nalazi zapadno od kule; *i* nije veća od kule; i ne postoji zgrada *j* među njima ($j > i$), koja je viša od zgrade *i*.



Pogledajte primer na slici gore: zgrade, koje mogu da da prime poruke, numerisane su brojevima 2, 5, 6, 9.

Napišite program **`tower`** da odredite maksimalni broj zgrada, koje mogu da prime saopštenja pri optimalnom položaju kula. Dat je poredak zgrada u gradu (u smislu datih visina zgrada) i visina kule.

Ulaz

U prvom redu standardnog ulaza data su dva, razmakom razdvojena, pozitivna cela broja: *N* i *H* – broj zgrada i visina kule.

U drugom redu standardnog ulaza dato je *N*, razmakom razdvojenih, pozitivnih celih brojeva – visine zgrada u gradu, uređene prema brojevima zgrada (od prvih do *N*-tih)

Izlaz

U jedinom redu standardnog izlaza štampajte jedan ceo broj – maksimalni broj zgrada koje će primiti saopštenja, ako je kula izgrađena pretpostavljajući njen optimalni položaj.

Ograničenja

$1 \leq N \leq 1\ 000\ 000$;

U 30% test slučajeva $N \leq 1000$;

$1 \leq$ visina svake zgrade i kule $\leq 10^9$

Primer

Ulaz	**Izlaz**
12 180 200 170 130 90 150 140 40 30 100 160 50 110	5

Pojašnjenje: Na slici ispod, data je optimalna lokacija kule. Saopštenja se primaju od strane zgrada sa brojevima 2, 5, 6, 7, 8.