Zadatak: "robot"

U nekoj se tvornici nalazi robotska ruka koja se kreće po proizvodnoj traci. Na traci se nalaze pakiranja proizvoda i ruka može u pojedino pakiranje dodati ili oduzeti predmet. Ruka se uvijek nalazi iznad točno jednog pakiranja (na početku iznad prvog), i to pakiranje nazivamo *ciljnim*.

Slijed takvih pakiranja zvat ćemo `traka`. Pretpostavljamo da je slijed pakiranja jednak kroz cijelo izvođenje programa. Međutim, broj predmeta u pojedinom pakiranju može se mijenjati.

Primjerice, ako je niz `brojevi` neki niz cijelih brojeva $\{4, 8, 1, 2\}$, tada sljedeći dio programa ispisuje `9 6 3 2`.

```
traka tr(brojevi, 4);
robot ro;

ro | new pozicija(2)
   | new uvecanje(2)
   | new pozicija(0)
   | new uvecanje(5)
   | new pozicija(1)
   | new uvecanje(-2);

ro(tr);

// ovdje operator modulo daje broj predmeta u pakiranju s danim indeksom
cout << tr % 0 << '␣' << tr % 1 << '␣' << tr % 2 << '␣' << tr % 3 << endl;
```

Opširniji testni klijentski program nalazi se na dnu zadatka.

Potrebno je implementirati strukture `traka`, `akcija`, `pozicija` (podtip od `akcija`), `uvecanje` (podtip od `akcija`) i `robot`.

Struktura `traka` pamti kolekciju cijelih brojeva (`int`) te informaciju o trenutno *ciljnom* pakiranju. Postoji konstruktor koji prima niz (polje) `int[]` i njegovu duljinu (`int`), i tako primljenom kolekcijom elemenata inicijalizira traku. Osim toga, `traka` definira i operator modulo (`operator%`) koji prima indeks te vraća referencu na broj spremljen u kolekciji na tom indeksu.

Struktura `akcija` pamti jedan cijeli broj (`int`). Taj broj zovemo argumentom akcije. Struktura `akcija` definira čisti virtualni `operator()` koji uzima jedan parametar: referencu na objekt tipa `traka`, i ništa ne vraća. Definira i operator pretvorbe u `int`. Taj operator vraća argument akcije.

Struktura `pozicija` je podtip strukture `akcija`. Nadograđuje virtualni `operator()` tako da on pomiče robotsku ruku na poziciju iznad pakiranja s indeksom argumenta akcije (varijabla članica strukture `akcija`). Mora postojati konstruktor koji uzima cijeli broj (`int`) i njime inicijalizira argument akcije.

Struktura `uvecanje` je također podtip strukture `akcija`. Ova struktura nadograđuje virtualni `operator()` tako da on mijenja broj predmeta u *ciljnom* pakiranju za (relativan) iznos argumenta akcije (varijabla članica strukture `akcija`). Mora postojati konstruktor koji uzima cijeli broj (`int`) i njime inicijalizira argument akcije. Osim toga, `uvecanje` definira i operator pridruživanja koji primljeni string (`std::string`) sastavljen od nepraznog niza znamenaka (iz skupa $\{0, \ldots, 9\}$) interpretira kao zapis nekog (nenegativnog cijelog) broja u sustavu s bazom 10 te tako pročitan broj sprema kao argument akcije. Taj operator vraća

referencu na objekt koji je odredište pridruživanja.

Struktura `robot` mora moći pamtiti pokazivače na akcije. Definira i `operator()` koji prima referencu na objekt tipa `traka` i slijedno izvršava sve robotske akcije. Operator ne vraća ništa pozivatelju. Osim toga, definira i `operator|` kojim se dodaje nova akcija. Taj operator prima objekt tipa `akcija*`.

Sučelje struktura koje definirate spremite u datoteku `robot.h`. Implementaciju spremite također u `robot.h`, ili u novu datoteku `robot.cpp`. Implementacijsku datoteku predajte čak i ako je prazna, tj. ako ste implementaciju napisali u datoteci `robot.h`.

Testni primjer:

```
#include <iostream>
#include "robot.h"

using namespace std;

int main() {
  int br[] = {3, 1, 4, 1};
  traka tr(br, 4);
  cout << tr % 0 << '␣' << tr % 1 << '␣' << tr % 2 << '␣' << tr % 3 << endl; /* izlaz: 3 1 4 1 */

  tr % 3 = tr % 2 = 6;
  cout << tr % 0 << '␣' << tr % 1 << '␣' << tr % 2 << '␣' << tr % 3 << endl; /* izlaz: 3 1 6 6 */

  robot ro;

  ro | new uvecanje(30) | new pozicija(1)
     | new pozicija(0) | new uvecanje(-1)
     | new pozicija(3) | new uvecanje(80);

  uvecanje pom(0);
  pom = "52";

  ro | new pozicija(1) | &pom;

  ro(tr);
  cout << int(pom) << '␣' << tr % 0 << '␣' << tr % 1 << '␣'
       << tr % 2 << '␣' << tr % 3 << endl; /* izlaz: 52 32 53 6 86 */

  return 0;
}
```

Napomene:

- Na pojedinom objektu tipa `robot` najviše će se jednom pozvati operator pozivanja (`operator()`). Pojedini objekt tipa `traka` bit će argument u najviše jednom takvom pozivu.
- Nigdje ne morate provjeravati smislenost parametara koji dolaze iz klijentskog dijela programa, npr. ne morate provjeravati jesu li indeksi u granicama.
- Pozivatelj nikad neće koristiti konstantne varijable bilo kojeg među spomenutim tipovima.
- Sve operatore možete definirati unutar definicija odgovarajućih struktura (ne morate koristiti globalne funkcije).
- U svojem rješenju nigdje ne morate dealocirati memoriju.

Ako imate dodatnih pitanja, javite na lmikec@math.hr