Napišite sučelje i implementaciju za strukture `vrt` i `gredica`. Struktura `vrt` predstavlja vrt pravokutnog oblika. Svaka gredica sadrži ime biljke koja se na njoj obrađuje (`string`), donji lijevi vrh te gornji desni vrh (zadani nenegativnim cjelobrojnim x i y koordinatama).
Struktura `vrt` predstavlja vrt koja sadrži kolekciju podataka tipa `gredica`. Ukupan broj gredica u vrtu nikad neće biti veći od 100. Gredice moraju u cijelosti biti sadržane unutar vrta i ne smiju se preklapati. Sučelje (deklaraciju) za obje strukture spremite u datoteku `vrt.h`, a implementaciju u datoteku `vrt.cpp`.
Strukture moraju imati sljedeće elemente (smijete dodati i elemente po želji, a neke ćete i morati dodati).

Struktura i funkcije	**Opis**
`gredica`	
`gredica(string biljka, int x1, int y1, int x2, int y2 )`	Stvara gredicu. Možete pretpostaviti da su koordinate točaka `(x1, y1)` i `(x2, y2)` dobro zadane tako da prva čini donji lijevi vrh gredice, a druga gornji desni (tj. `x1 <= x2`, `y1 <= y2`)
`string getBiljka()`	Vraća ime biljke koja se na gredici obrađuje.
`int povrsina()`	Vraća površinu gredice.
`vrt`	
`vrt(int x, int y)`	Stvara vrt koje u sebi nema gredica. Donji lijevi vrh vrta ima koordinate `(0,0)`, a gornji desni `(x,y)`.
`int brojgredica()`	Vraća broj gredica
`gredica getGredica(int index)`	`index` je broj između `0` i ukupnog broja gredica -1. Vraća određenu gredicu prema redoslijedu dodavanja u kolekciju
`bool dodajGredicu(gredica g)`	Dodaje gredicu kolekciji, ukoliko u kolekciji ima manje od 100 gredica, i ako su koordinate gredice takve da ne izlaze izvan okvira vrta i ako se gredica ne preklapa sa nekom drugom gredicom vraća `true`, a `false` inače (pogledajte primjer dolje).
`bool ukloniGredicu(int x, int y)`	Uklanja prvu gredicu na koju naiđe u kolekciji (po redoslijedu dodavanja) ako gredica sadrži točku `(x,y)`, i vraća `true`, a `false` inače.

`vrt sadrzi(string biljka)`	Vraća vrt iste veličine kao početni koji se sastoji samo od gredica na kojima je posađena biljka `biljka`. Gredice trebaju biti u onom redoslijedu u kakvom su bile u početnom vrtu.
`int promijeniVelicinu(int x1, int y1, int x2, int y2)`	Mijenja veličinu vrta tako da mu donja lijeva točka ima koordinate `(x1, y1)`, a gornja desna `(x2, y2)`. Također treba ukloniti sve gredice koje u cijelosti ne leže u novom okviru. Funkcija treba vratiti broj uklonjenih gredica.

Primjer klijentskog programa

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

using namespace std;


void ispisigredice(vrt v) {
	for (int i = 0; i < v.brojgredica(); ++i) {
		gredica g = v.getGredica(i);
		cout<<g.getBiljka()<<": "<<g.povrsina()<<" m2"<<endl;
	}
}


int main() {
	vrt v(10, 10);
	cout << v.dodajGredicu(gredica("kamilica", 0, 0, 3, 3)); // 1
	cout << v.dodajGredicu(gredica("mazuran", 3, 0, 10, 1)); // 1
	cout << v.dodajGredicu(gredica("metvica", 3, 2, 5, 4)); // 1
	cout << v.dodajGredicu(gredica("maticnjak", 5, 2, 7, 4)); // 1
	cout << v.dodajGredicu(gredica("aloa", 5, 6, 7, 8)); // 1
	cout << v.dodajGredicu(gredica("kim", 7, 2, 8, 5)); // 1
	cout << v.dodajGredicu(gredica("smilje", 1, 0, 6, 6)); // 0 - preklapa
se s postojećom gredicom
	cout << endl;

	ispisigredice(v);
	/*
	kamilica: 9 m2
	mazuran: 7 m2
	metvica: 4 m2
	maticnjak: 4 m2
	aloa: 4 m2
	kim: 3 m2
    */

	cout << v.brojgredica() << endl; // 6
	v.ukloniGredicu(4, 4); // postoji
	v.ukloniGredicu(6, 5); // ne postoji
	cout << v.brojgredica() << endl; // 5
```

```
      ispisigredice(v.sadrzi("kamilica"));
      /*
    kamilica: 9 m2
    */

      ispisigredice(v.sadrzi("sipak")); // nista

      cout << v.promijeniVelicinu(0, 0, 9, 5) << endl; // 2
    ispisigredice(v);
      /*
      kamilica: 9 m2
      maticnjak: 4 m2
      kim: 3 m2
    */
      cout << v.promijeniVelicinu(2, 2, 7, 4) << endl; // 2
    ispisigredice(v);
      /*
    maticnjak: 4 m2
    */
      cout << v.dodajGredicu(gredica("kadifa", 1, 1, 3, 4)); // 0 - izvan
granica vrta
      cout << v.dodajGredicu(gredica("preslica", 2, 2, 5, 4)); // 1
      cout << endl;

      ispisigredice(v);
      /*
      maticnjak: 4 m2
      preslica: 6 m2
    */

      return 0;
}
```

Opće napomene

• Struktura, funkcije i datoteke koje šaljete moraju se zvati *točno* onako kako je zadano u zadatku. Pazite na mala i velika slova!

• Trebate poslati samo sučelje i implementaciju. U datotekama koje šaljete *ne smije* se nalaziti funkcija `main()`!

• Nijedna funkcija *ne smije* ništa učitavati s tipkovnice ili neke datoteke, niti išta ispisivati na ekran ili u neku datoteku.

• Svaki od main-ova pomoću kojih testiramo ispravnost vašeg programa neće pozivati sve gore navedene funkcije. Stoga, ako neku od funkcija ne znate napisati ipak možete dobiti koji bod (u tom slučaju tu funkciju nemojte navesti niti u `.h` niti u `.cpp` datoteci ili napravite neku trivijalnu implementaciju).

Ispravnost implementacijâ koje napišete bit će provjerena tako da ćemo mi napisati razne klijentske programe koji će deklarirati nekoliko varijabli zadane strukture, i na njima pozivati funkcije koje ste trebali napisati. Ako se poslani programi ne budu uspješno povezivali (*linkali*) s našim klijentskim programima, smatrat će se neispravnima. Neki klijentski programi provjeravat će samo neke jednostavnije funkcije, dok će neki provjeravati sve funkcije koje trebate napisati. Provjera je potpuno automatska, tako da je od presudne važnosti da se pridržavate specifikacije. Nepridržavanje lako može uzrokovati osvojenih 0 bodova iz zadaće! Naravno, za provjeru radi li implementacija

prije nego što je pošaljete, preporučuje se da je testirate pomoću nekog klijentskog programa. No taj klijentski program ne šaljete!