

به نام خدا

سوال اول

کلاسی به نام `unique pairs` برای ذخیره سازی مجموعه ای نامحدود از داده ها طراحی کنید که داده ها در آن به صورت زوج های مرتب (کلید، مقدار) هستند. در واقع در این مجموعه، کلید نقش اندیس را بازی می کند و می توان بر اساس کلید به عنصر مقدار دسترسی پیدا کرد. قوانینی برای این مجموعه وجود دارد:

- کلید منحصر به فرد است، یعنی امکان ندارد دو زوج با کلید یکسان در مجموعه وجود داشته باشد. نوع کلید-و نوع مقدار برای همه عناصر یک مجموعه واحد است. یعنی مثلاً یک مجموعه داریم که کلید همه عناصر از نوع `int` و مقدار همه عناصر از نوع `double` است. اما می توان مجموعه ی دیگری با نوع کلید `float` و مقدار `string` هم داشت.

نیاز به مرتب کردن داده ها بر حسب کلید نیست.

این کلاس باید حاوی امکانات زیر باشد:

- `Insert`: یک زوج داده ای را دریافت کرده و در صورت یکتا بودن کلید، آن را به مجموعه اضافه کند. در صورت تکراری بودن کلید، زوج جدید جایگزین زوج قبلی می شود.
- `Remove`: یک کلید را دریافت کرده و در صورت وجود در مجموعه، زوج داده ای آن را حذف می کند.
- تابع `size`: تعداد زوج های موجود در مجموعه را نشان می دهد.
- اپراتور چاپ: همه داده های مجموعه به صورت زوج (کلید، مقدار) یکی یکی چاپ شوند.
- اپراتور `[]`: امکان دسترسی به فیلد مقدار با کمک کلید. در صورتی که آن کلید وجود نداشته باشد، یک زوج جدید با آن کلید و مقدار پیش فرض ایجاد می شود.
- اپراتور `-`: تفاضل دو مجموعه را محاسبه می کند. یعنی `A-B` برابر مجموعه `A` بدون زوج های (کلید، مقدار) مشترک با مجموعه `B` می شود.

کلاس دیگری به نام `pairs` طراحی کنید که مانند کلاس قبل شامل زوج های (کلید، مقدار) باشد، اما در این کلاس شرط یکتا بودن کلید وجود ندارد. امکانات این کلاس به شرح زیر است:

- `Insert`: یک زوج داده ای را دریافت کرده و آن را به مجموعه اضافه کند. در صورت تکراری بودن زوج (کلید، مقدار) زوج جدید اضافه نمی شود.
- `Remove`: یک کلید را دریافت کرده و در صورت وجود در مجموعه، همه زوج داده های آن را حذف می کند.

- تابع size: تعداد زوج های موجود در مجموعه را نشان می دهد.
- اپراتور چاپ: مجموعه داده های مجموعه به صورت زوج (کلید، مقدار) یکی یکی چاپ شوند.
- اپراتور - : تفاضل دو مجموعه را محاسبه می کند. یعنی $A-B$ برابر مجموعه A بدون زوج های (کلید، مقدار) مشترک با مجموعه B می شود.

کلاس های خود را به گونه ای طراحی کنید که با استفاده از آن ها بتوان کد زیر را اجرا کرد. در طراحی کلاس ها از امکانات شی گرایی به گونه ای استفاده کنید که کد شما حداقل باشد.

```
int main()
{
    uniquePairs<char, int> first;

    first['x'] = 8;    //first= {('x',8)}
    first['y'] = 16;   //first= {('x',8), ('y',16)}
    first['z'] = 32;   //first= {('x',8), ('y',16), ('z',32)}

    first.insert('a', 100); //first= {('x',8), ('y',16), ('z',32), ('a',100)}
    first.insert('x', 200); //first= {('x',200), ('y',16), ('z',32), ('a',100)}
    first.remove('x');    //first= {('y',16), ('z',32), ('a',100)}
    first.remove('a');    //first= {('y',16), ('z',32)}

    uniquePairs<char, int> second;
    second = first;       // second={('y',16), ('z',32)}

    cout << second;      //console output: {('y',16), ('z',32)}

    std::cout << "Size of first: " << first.size() << '\n'; //output: 2
    std::cout << "Size of second: " << second.size() << '\n'; //output: 2

    second.insert('b', 40);
    second.insert('x', 70);
    second.insert('a', 100); // second={('y',16), ('z',32), ('b',40), ('x',70), ('a',100)}
    second = second - first; //second= ( ('b',40), ('x',70), ('a',100)}

    first.insert('x', 8);
    const uniquePairs<char, int> third = first;

    second['a']=third['y'];
    cout << third;

    pairs<string, string> phoneBook1;
    phoneBook1.insert("ali", "123");
    phoneBook1.insert("ali", "456");
    phoneBook1.insert("mina", "5678");

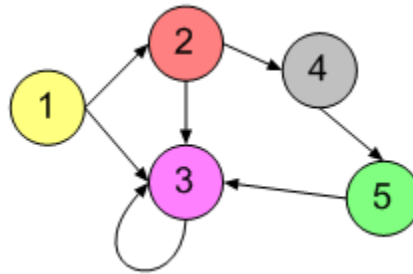
    pairs<string, string> phoneBook2;
    phoneBook2["ahmad"] = "1234567";
    phoneBook2["mina"] = "1234567";
```

```
phoneBook2["mina"] = "567";  
const pairs<string, string>phoneBook3 = phoneBook2;  
phoneBook1.remove("ali");    // phoneBook1 = { ("mina", "567") }  
cout << phoneBook3;  
  
return 0;  
}
```

به نام خدا

سوال دوم

یکی از مفاهیم کاربردی در علم ریاضی و کامپیوتر گراف است. برای مدیریت گراف جهت دار کلاسی طراحی کنید به نحوی که بتوان گراف‌هایی با نوع‌های مختلف را مدیریت کرد، مثلاً گرافی از مقادیر صحیح یا گرافی از مقادیر اعشاری. در یک گراف هر گره حاوی یک مقدار بوده و یال‌هایی به یک یا چند گره دارد. دقت کنید که در یک گراف هیچ دو گره نباید مقدار یکسان داشته باشند و در عین حال مقدار گره‌های یک گراف همه از یک نوع داده‌ای هستند. شکل زیر یک نمونه گراف جهت دار را با مقادیری از نوع `int` نشان می‌دهد. برای سادگی می‌توانید فرض کنید که هر گراف حداکثر n گره دارد، که مقدار n در سازنده دریافت می‌شود.



کلاسی که طراحی می‌کنید باید دارای قابلیت‌های زیر باشد:

- افزودن گره: با دریافت مقدار داده‌ای یک گره و لیست گره‌های همسایه‌ی آن، گره جدید به گراف اضافه شود. مثلاً در شکل بالا برای افزودن گره ۲ به گراف، مقادیر ۲ (مقدار نود) و ۱:۲ و ۳:۲ به عنوان یال‌های گره به تابع داده می‌شود. هر گره حداقل باید به یک گره دیگر لینک داشته باشد.
- افزودن یال: با دریافت آدرس دو گره، می‌توان یالی از یک گره به گره دیگر اضافه کرد.
- حذف یال: با دریافت آدرس دو گره و جهت یال، می‌توان آن یال را حذف کرد.
- حذف گره: با دریافت مقدار داده‌ای یک گره، آن گره حذف شود. با حذف یک گره باید لینک‌های آن به سایر گره‌ها حذف شود. همچنین، لینک‌های سایر گره‌ها به این گره حذف گردد.
- حذف گره با دریافت آدرس گره: با دریافت اشاره‌گری به یک گره، آن را از گراف حذف می‌کند. با حذف یک گره باید لینک‌های آن به سایر گره‌ها حذف شود. همچنین، لینک‌های سایر گره‌ها به این گره حذف گردد.
- جستجوی گره: با دریافت یک مقدار، در گراف جستجو کرده و در صورت وجود اشاره‌گری به گره حاوی آن مقدار بر می‌گرداند. در غیر اینصورت `NULL` بر می‌گرداند.
- اپراتور چاپ: باید یکی یکی گره‌های گراف و لیست گره‌هایی که به آن‌ها یالی خروجی دارد، چاپ شود.

کلاس خود را به گونه‌ای طراحی کنید که کد زیر قابل اجرا باشد. همچنین، طراحی شما باید به گونه‌ای باشد که کد شما حداقل باشد. در این برنامه محدودیت زمان اجرا و حافظه وجود ندارد، اما طراحی که میزان حافظه مصرفی را حداقل کند، امتیاز تشویقی خواهد داشت.

```
graph<int,10> a;  
//insert nodes to graph a  
graph<int,10>b = a;
```

```
//insert to or remove nodes from graph b
graph <int,10> c;
c = b;
cout << b;
auto x = b.find(12);
if (x)
    b.remove(12);
```