Union

$A = \{1,2,3\}$

$B = \{3,4\}$

$A \cup B = \{1,2,3,4\}$ $\qquad$ $A \cap B = \{3\}$

```
bool    exists (vector <int> v, int q)
{
        for (int i=0;  i < v.size();  ++i) {
                if (v[i] == q) {
                        return true;
                }
        }
        return false;
}
```

```
void  union1(vector <int>& a, vector<int>& b)
{
    vector <int>  c;
    for (int  i=0;  i< a.size();  ++i) {
          c.push_back (a[i]);
    }
    for (int  i=0;  i< b.size ();  ++ i) {
          if (exists (a, b[i]))  {
                continue;
          }
          else   {
                c.push_back( b[i]);
```

```
            }
        }
    }

void  intersect (vector <int> a,  vector <int> b)
{
    vector <int> c;
    for (int  i=0;  i< a.size();  ++i ) {
          if ( exists (b, a[i])) {
               c.push_back (a[i]) ;
          }
    }
}
```

```
        0 1 2 3 4 5 6 7 8 9 10 11 12 11 12 13 14 15 16
Str:    i   a m   n o t   s u r e   o f   i t          -> 17
```

query: am~~o~~

output: yes, 2.

"m not sure"

"am no"   "am"   2

```
bool  match (char* s,  char* q , int l)
{
     for (int  i=0;  i<l;  ++i)  {
          if (s[i]  != q[i])  {
               return  false;
          }
```

```
        }
        return true;
    }
```

s = "This is so great"]
q = "This is so"] T

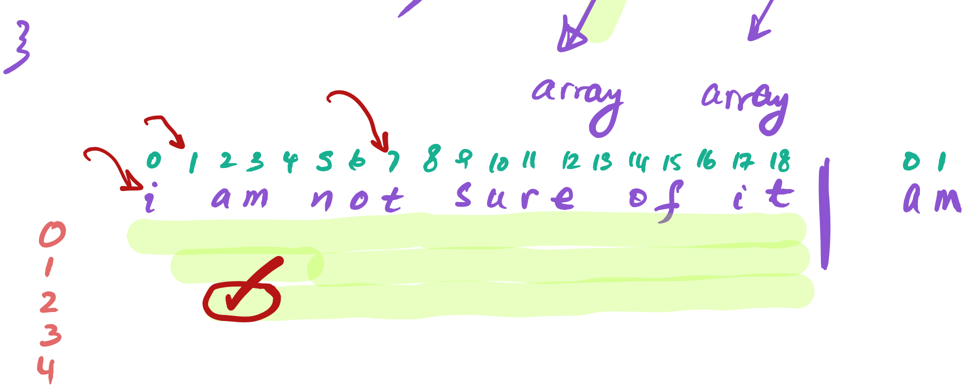
q = "Th"] T

q = "h"] F

q = "This is crazy"] F

```
                                     *         *
int find(char *s, int lens, char *q, int leng)
{     vector<int> pos;
      for(int i=0; i< lens-leng ; ++i) {
          if(matches(&s[i], q, leng)) {
                  pos.push_back(i);
          →       ~~return~~ i;
          }
      }
      return -1;
}
```

s[i]



```
5
6     s[0] = 'i'          & s[0]    |   s     } Pointer
7                                   |   s+1   } Arithmetic
8     s[7] = 't'          & s[1]    |   s+7
9                         & s[7]
10
11
12
13
14
15
16
```

```
         0 1 2 3 4 5 6 7 8 9 10 11 12 13
str :    a b b c d e f f f f d  3  x  4      —— 14
```

```
void  ss(char* s, int lens, char& c, int& p, int & l)
{
          current
     char c1 = c = s[0];   -> current best
     int  p1 = p = 0;
     int  l1 = l = 1;

     for (int i=1; i< lens ; ++i)   {
          if ( s[i] == c1 )  {
               ++l1;
               if (l1 > l)  {
                    c = c1;
                    p = p1;   -> current best.
                    l = l1;
               }
          }
          else  {
               c1 = s[i];
               p1 = i;
               l1 = 1;
          }
}
```

}

Write a class to store the name and address of an individual.

name
address

```
...
int main(int argc, char** argv)
{
    Person p("Jhon");
    p.set_address("2000 Simcoe ...");
    cout << p << endl;

    Person q(p);

    cout << (p == q) << endl;
    ...
}
```

```
class Person {
  protected:
      string name;
      string addr;
  public:
      Person()  {}
      Person(const string& name)  {
          this->name = name;
```

```
        }
        Person (const Person & p) {
                name = p.name;
                addr = p.addr;
        }
        void set_address(const string & addr) {
                this -> addr = addr;
        }
        friend ostream& operator<<(ostream& os, const Person&
                                                        p);

        bool operator==(const Person& p) {
            return (name == p.name && addr == p.addr);
        }
};

ostream & operator<<(ostream& os, const Person& p)
{
            os << p.name << endl;
            os << p.addr;
            return os;
}
```