EXAMPLE E.28 Testing the `merge()` Algorithm

```
int main()
{ int a[] = {22,55,66,88};
  int b[] = {11,33,44,77,99};
  int c[9];
  merge(a,a+4,b,b+5,c);
  print(c,9);
}
n=9: {11,22,33,44,55,66,77,88,99}
```

`min(x,y);`
```
// returns the minimum of x and y;
```

EXAMPLE E.29 Testing the `min()` Algorithm

```
int main()
{ cout << "min(48,84)=" << min(48,84) << '\n';
}
min(48,84)=48
```

`min_element(p,q);`
```
// returns the position of the minimum element in the segment [p,q[;
// invariant: [p,q[ is left unchanged;
```

EXAMPLE E.30 Testing the `min_element()` Algorithm

```
int main()
{ int a[] = {77,22,99,55,11,88,44,33,66};
  const int* p = min_element(a,a+9);
  cout << "*p=" << *p << '\n';
  cout << "p-a=" << p-a << '\n';
}
*p=11
p-a=4
```

`mismatch(p,q,pp);`
```
// returns a pair of iterators giving the positions in [p,q[ and
// in [pp,qq[ where the first mismatch of elements occurs;
// if the two segments match entirely, then their ends are returned;
// invariant: [p,q[ and [pp,qq[ are left unchanged;
```

EXAMPLE E.31 Testing the `mismatch()` Algorithm

```
int main()
{ char* s1="Aphrodite, Apollo, Ares, Artemis, Athena";
  char* s2="Aphrodite, Apallo, Ares, Artimis, Athens";
  int n=strlen(s1);
  cout << "n=" << n << '\n';
```