2.6

```
template<class T>
void Reverse(T A[], int n, int arraySize) {
    if (n > arraySize) {
        std::cerr << "Error: n is greater than array size" << std::endl;
        return;
    }
    T temp;
    for (int i = 0; i < n / 2; i++) {
        temp = A[i];
        A[i] = A[n - i - 1];
        A[n - i - 1] = temp;
    }
}
```

2.11

```
template<class T>
void FindMinandMax(SeqList<int>& L, int& min, int& max)
{
    int n = L.Length(), x;
    L.getData(1, max);
    min = max;
    for (int i = 2; i <= n; i++)
    {
        L.getData(i, x);
        if (x > max) max = x;
        else if (x < min) min = x;
    }
}
```

2.12

```
SeqList<int> Merge(SeqList<int>& A, SeqList<int>& B) {
    SeqList<int> C;
    int i = 1, j = 1;
    int valA, valB;
    A.getData(i, valA); B.getData(j, valB);
    while (i <= A.Length() && j <= B.Length()) {
        if (valA <= valB) {
            C.Append(valA);
            i++;
            A.getData(i, valA);
        }
        else {
            C.Append(valB);
            j++;
            B.getData(j, valB);
        }
```

```
    }
    while (i <= A.Length()) {
        C.Append(valA);
        i++;
        A.getData(i, valA);
    }
    while (j <= B.Length()) {
        C.Append(valB);
        j++;
        B.getData(j, valB);
    }
    return C;
}
```

2.16

(1)

```
template<class T>
void toRight(List<T>& L, LinkNode<T>*& p, LinkNode<T>*& pr, int k) {
    LinkNode<T>* q;
    int count = 0;
    while (count < k && p != NULL) {
        q = p->link;
        p->link = pr; pr = p; p = q;
        count++;
    }
    if (p == NULL) {
        L.setTail(L.First()->link);
        L.First()->link = pr;
    }
}
```

(2)

```
template<class T>
void toLeft(List<T>& L, LinkNode<T>*& p, LinkNode<T>*& pr, int k) {
    LinkNode<T>* q;
    q = p; p = pr; pr = q;
    int count = 0;
    while (count < k && p != NULL) {
        q = p->link;
        p->link = pr; pr = p; p = q;
        count++;
    }
    if (p == NULL) {
        L.setTail(L.First()->link);
        L.First()->link = pr;
    }
}
```

2.17

```
template<class T>
void Reverse(LinkNode<T>*& h) {
    if (h == NULL) return;
    LinkNode<T>* p = h->link, * pr = 0;
    while (p != NULL) {
        h->link = pr;
        pr = h; h = p; p = p->link;
    }
    h->link = pr;
}
```

2.19

```
template<class T>
void LinkOrder(DblLink<T>& DL) {
    DblNode<T>* pr, * p, * s, * h;
    h = DL.First();
    s = h->rLink->rLink;
    h->lLink = h->rLink; h->rLink->lLink = h;
    while (s != h) {
        p = h->lLink; pr = h;
        while (p != h && p->data < s->data) {
            pr = p; p = p->lLink;
        }
        pr->lLink = s; s->lLink = p;
        s = s->rLink;
    }
}
```