Exercises 7

张逸松

October 27, 2019

7.2

```
stack insertSort(stack & in) {
    stack ret, tmp;
    for (; !in.empty(); ) {
        for (; !ret.empty() && ret.top() > in.top();tmp.push(ret.top()),
    ret.pop());
        ret.push(in.top()); in.pop();
        for (; !tmp.empty(); tmp.pop()) ret.push(tmp.top());
    }
    return ret;
}
```

7.12

```
inline int findpivot(int i, int j) {
    return (i + j) >> 1;
}
inline void swap(string *&prt[], int a, int b) {
    string *tmp = prt[a];
    prt[a] = prt[b];
    prt[b] = tmp;
}
inline bool prior(string *a, string *b) {
    if (*a.size() != *b.size()) return *a.size() < *b.size();
    for (int i = 0; i <= *a.size(); i++)
        if (*a[i] != *b[i]) return *a[i] < *b[i];
    return 0;
}
inline int partition(string *ptr[], int l, int r, string * pivot) {
    do {
        while (prior(ptr[++l], pivot));
        while ((l < r) && prior(pivot, ptr[--r]));
        swap(A, l, r);
    } while (l < r);
    return l;
}
void qsort(string *ptr[], int i, int j) {
    if (j <= i) return;
    int pivotindex = findpivot(A, i, j);
    swap(ptr, pivotindex, j);
    int k = partition (ptr, i - 1, j, ptr[j]);
    swap(ptr, k, j);
    qsort(ptr, i, k - 1);
    qsort(ptr, k + 1, j);

}
```

7.20

```
LList <int> mergeSort(LList <int> in) {
    if (in.length() == 1) return in;
    in.setStart();
    LList <int> tmp[2];
    for (int cur = 0; !in.isEmpty(); cur ^= 1) tmp[cur].append(in.remove
   ());
    mergeSort(tmp[0]);
    mergeSort(tmp[1]);
    tmp[0].setStart();
    tmp[1].setStart();
    for (;!tmp[0].isEmpty() && !tmp[1].isEmpty(); ) {
        if (tmp[0].head->element < tmp[1].head->element) in.append(tmp
   [0].remove());
        else in.append(tmp[1].remove());
    }
    if (!tmp[0].empty()) in.append(tmp[0].remove());
    if (!tmp[1].empty()) in.append(tmp[1].remove());
    return in;
}
```