```
1  /*-------------------------------------------------------------------
2  Copyright (c) 2014 Author: Jagadeesh Vasudevamurthy
3  file: dsort.hpp
4
5  -------------------------------------------------------------------*/
6
7  /*-------------------------------------------------------------------
8  This file has class definition
9  -------------------------------------------------------------------*/
10
11 /*-------------------------------------------------------------------
12 Definition of routines of dsort class
13 -------------------------------------------------------------------*/
14
15
16 /*-------------------------------------------------------------------
17 Constructor
18 -------------------------------------------------------------------*/
19 template <typename T>
20 dsort<T>::dsort(darray<T>& d, int(*cf) (const T& c1, const T& c2), int size) :_darray(d), _cf(cf),      ↙
       _size(size){
21
22 }
23
24 /*-------------------------------------------------------------------
25 Destructor
26 -------------------------------------------------------------------*/
27 template <typename T>
28 dsort<T>::~dsort() {
29
30 }
31
32 /*-------------------------------------------------------------------
33 print darray
34 -------------------------------------------------------------------*/
35 template <typename T>
36 void dsort<T>::_print_darray(int pass){
37   cout << pass << ":";
38   for (int i = 0; i < _size; i++) {
39     cout << _darray[i] << " ";
40   }
41   cout << endl;
42 }
43
44 /*-------------------------------------------------------------------
45 Swap two object in the array
46 By value:      Object must have = operator
47 By ptr:       Pointers are compared
48 -------------------------------------------------------------------*/
49 template <class T>
50 void dsort<T>::_swap(T& a, T& b) {
51   T temp = a;
52   a = b;
53   b = temp;
54 }
55
56 /*-------------------------------------------------------------------
57 Binary search
58
59 0  1  2 3 4 5 6 7 8 9  10 11
60 -5 -3 -1 0 1 4 7 7 9 10 21 45
61
62 NOTION HERE: first = 0 ;
63 last = 11;  //NOT 12. Last is index to Largest element
64
65
```

```
-4 Not there in array. But can be put in position 1 (false)

21 there in position 10 (true)

46 Not there in array. But can be put in position 12 (false)

-8 Not there in array. But can be put in position 0 (false)

pos gives the position where r can be added.
In the above case pos can be 0 or pos can be 12
-----------------------------------------------------------------*/
template <typename T>
bool dsort<T>::binary_search(const T& r, int first, int last, int& pos) {
  assert(first <= last);
  int k;
  int middle = 0;
  /* there must be at least one element: a[0] */
  while (first <= last) { /* Even one element  */
    middle = (first + last) / 2;
    k = _cf(_darray[middle], r);
    if (k == 0) {
      //equal
      pos = middle; //Solution found in middle
      return true;
    }
    if (k < 0) {
      //r is lesser than array[middle]
      last = middle - 1;
    }
    else {
      //r is greater than array[middle]
      first = middle + 1;
    }
  }
  //middle is the index above or below target
  if (last < 0) {
    pos = 0; //0
  }
  if (first > last) {
    pos = first; //n+1 case
  }
  assert(pos >= 0 && pos <= last + 1); //can be added at 0 or last +1
  return false;
}

/*-----------------------------------------------------------------
prove that the array is really sorted
-----------------------------------------------------------------*/
template <typename T>
void dsort<T>::assertSorted(){
  int p = 0;
  for (int i = 1; i < _size; ++i) {
    int r = _cf(_darray[p], _darray[i]);
    assert(r >= 0);
    p = i;
  }
}

/*-----------------------------------------------------------------
Bubble sort

0:25 57 48 37 12 92 86 33
1:25 48 37 12 57 86 33 92
2:25 37 12 48 57 33 86 92
3:25 12 37 48 33 57 86 92
4:12 25 37 33 48 57 86 92
```

```
5:12 25 33 37 48 57 86 92
6:12 25 33 37 48 57 86 92
-----------------------------------------------------------------*/
template <typename T>
void dsort<T>::bubble_sort(){
  reset_stat();
  if (display()) {
    _print_darray(0);
  }
  bool exchanged = false;
  int n = _size;
  do{
    inc_num_iteration();
    exchanged = false;
    for (int i = 0; i < n - 1; i++) {
      int j = i + 1;
      inc_num_compare();
      if (_cf(_darray[i], _darray[j]) < 0) {
        inc_num_swap();
        exchanged = true;
        _swap(_darray[i], _darray[j]);
      }
    }
    if (display()) {
      _print_darray(get_num_iteration());
    }
    n--;
  } while (exchanged);
  assertSorted();
  cout << "----------Bubble sort-------------------\n";
  cout << "#n =               " << _size << endl;
  int c = get_num_compare();
  int s = get_num_swap();
  int it = get_num_iteration();
  int t = c + s;
  cout << "#comparison =      " << c << endl;
  cout << "#swap        =     " << s << endl;
  cout << "#Bubbled loops =   " << it << endl;
  cout << "#c+s =             " << t << endl;
  double x = double(t) / double(_size*_size);
  cout << "#T(n) = (c+s)/(n^2) =  " << x << "(n^2)" << endl;
  cout << "----------------------------------------\n";
}

/*-----------------------------------------------------------------
Insertion sort
-----------------------------------------------------------------*/
template <typename T>
void dsort<T>::insertion_sort(){
  reset_stat();
  cout << "WRITE YOUR CODE HERE\n";

  //assertSorted();
  //WRITE statistics code
}


/*-----------------------------------------------------------------
Merge sort
-----------------------------------------------------------------*/
template <typename T>
void dsort<T>::merge_sort(){
  reset_stat();
  cout << "WRITE YOUR CODE HERE\n";

  //assertSorted();
```

```
198   //WRITE statistics code
199 }
200
201 /*-------------------------------------------------------------------
202 Quick sort
203
204 -------------------------------------------------------------------*/
205 template <typename T>
206 void dsort<T>::quick_sort(){
207   reset_stat();
208   cout << "WRITE YOUR CODE HERE\n";
209
210
211   //assertSorted();
212   //WRITE statistics code
213 }
214
215 //EOF
216
217
```