Assignment 14 (sorted set and recursion)

The three functions you have to write (either to replace or add new) for Assignment 14 are `isetAdd()`, `isetRemove()`, and `isetIndexFast()`.

Sorted Order

Step a) Change the implementation of `isetRemove` so that it does a stable remove. In a stable remove, all values in the list with an index greater than the index of the value to be removed get moved to the next lower index. Make sure you do not move or otherwise touch any values that are not in the list (i.e. at `length` or before index 0).

Stable remove, requires a for loop. If you do it right, it should only need two tests, one of which is the usual `i` (or `index`) `< length`. Remember, the buffer is already in sorted order, so you don't have to test for being "between" (if you passed an index, you should already know that the new value is greater than the value at the index you just passed, so don't test it again). Also, if you do it right, the loop should exit when `i` (or `index`) is the correct location for insertion, even if the new value goes at the front or end of the existing list. You can combine steps a) and b) if you stop on the value (and not just the first value greater, and then test the index where it stops to see if the value is already there.

Step b) Change the implementation of `isetAdd` so that if it adds a new value to the buffer, it preserves the property that the buffer is in sorted order. That means you have to a) locate the correct index where the new value should be inserted (at the beginning, between two existing values, or at the end). Then b) move every value at that index or higher to the next higher index in the buffer. Finally, c) assign the new value in its chosen location. You also had to change the value of `length`.

Ordered add, also requires a loop. In this case the loop must move values from the chosen index on up to the next higher index. Note that you cannot perform this move (right shift) in the same order you did the move for delete (left shift). Doing this in the wrong order will result in all values in the upper part of your list being the same. The for loop for shifting left has to start at the end and work down towards the index.

In both loops, you have to be careful to start and end the loop in the right place. Your copy should not involve moving a value that is beyond the end of the array, or moving a value at the beginning into a location that gets replaced by a new value.

A good way to implement both of these functions is to create separate "helper" functions called `shiftUpOne` and `shiftDownOne`. As helper functions these functions should be declared with a `static` prefix. Putting the word `static` at the beginning of the declaration of a function tells the compile that this function is only for use within the same file, and cannot be called from code in other files (like from the hw14.c file).

DO NOT IMPLEMENT AN `isetSort` FUNCTION. DOING SO INDICATES THAT YOU DO NOT UNDERSTAND THIS ASSIGNMENT. (There should never be a situation where the buffer is not already sorted.)

For one point of extra credit:

Add a new function called `isetIndexFast`. In this case, you will use binary search to find the index, in place of the original linear search. In binary search instead of starting at the

beginning, you start you search in the middle. Divide `length` by 2 (or left shift 1) to get the halfway index. If you found the value, return that index. If not, test for value for being less than the value at that index, in which case you search the lower half, else you search the upper half. Repeat the above sequence until the value is found, or there is nothing left to search (the lower index is bigger than the upper index). If there is nothing left to search, return -1.

isetIndexFast

By default, the `isetIndexFast` code is not tested in hw14.c. If you implement this part, use:

```
gcc –DFAST hw14.c IntegerSet.c
```

If you have done this part and you used a loop, redo it as a recursive function. The recursive solution is easier to write and will demonstrate the things you are supposed to learn from in the curriculum for this course.

To use a recursive function, you must write a fifth function specifically for the recursion. The `isetIndexFast` function itself cannot be used as the recursive function. Instead, it should call the recursive "helper" function to get it started (searching the entire span from 0 to length – 1) and return what it returns. Here is what you should have in your file: You need to write nine lines of the `recursiveFind` function. The `recursiveFind` should be declared with the word `static` in front, which makes is usable only from within the file where it appears. It is only to be used by `isetIndexFast`, and therefore should not be called from anywhere else.

Here is a prototype and skeleton for the `recursiveFind` function, and the entire implementation of `isetIndexFast`.

```
static int recursiveFind(const int array[], int value,
                         int lowIndex, int highIndex);

int isetIndexFast(const IntegerSet *iset, int value) {
    return recursiveFind(iset->buffer, value, 0, iset->length – 1);
}

static int fastFind(const int array[], int value,
                    int lowIndex, int highIndex) {
    /* compute the middle index */

    /* not found: if the lowIndex is higher than the highIndex,
     * return -1 */

    /* found: if value is same as array[middleIndex],
     * return middleIndex */

    /* search low: if value is less than array[middleIndex],
     * return the result of calling recursiveFind with the lower half
     * (between lowIndex and middleIndex – 1) */

    /* search high: else (value is greater than array[middleIndex]
     * return the result of calling recursiveFind with the upper half
     * (between middleIndex + 1 and highIndex */

}
```

Here is the expected output from hw14.c, including the fast part:

```
% a.out
Phase 1: test that isetCreate is working correctly
size: 35, length: 0
Phase 2: test that isetAdd is working correctly
      2      5     17
Phase 3: test that isetAddArray is working correctly
      2      5     17     22     23     24     25     26     27     28
     29     30     31     32     34     35     36     38     39     40
     41     42     43     44     45     46     48     49     50
Phase 4: test that isetAdd is working with negative numbers
    -24    -19    -18     -9     -8     -4      2      5     17     22
     23     24     25     26     27     28     29     30     31     32
     34     35     36     38     39     40     41     42     43     44
     45     46     48     49     50
Phase 7: test that isetClonee is working correctly (part 1)
Phase 5: test that isetClear is working correctly
      0      1      2      3      4      5
Phase 6: test that isetIndex, isetRIndex, and isetRemove work
correctly
      0      1      3      5
      1      3
Phase 7: test that isetClone is working correctly (part 2)
    -24    -19    -18     -9     -8     -4      2      5     17     22
     23     24     25     26     27     28     29     30     31     32
     34     35     36     38     39     40     41     42     43     44
     45     46     48     49     50
Phase 8: test that isetResize is working
    -24    -19    -18     -9     -8     -4      2      5     17     22
     23     24     25     26     27     28     29     30     31     32
     34     35     36     38     39     40     41     42     43     44
     45     46     48     49     50
Phase 9: test that isetResize is working correctly
    -24    -19    -18     -9     -8     -4      2      5     17     22
     23     24     25     26     27     28     29     30     31     32
     33     34     35     36     38     39     40     41     42     43
     44     45     46     48     49     50     58     63     69     71
     77     82     87     88     97    107    112    114    116    121
    124    125    128    138    139    140    142    149    153    154
    158    161    172    174    175    181    185    198    201    202
    204    207    209    214    220    227    229    231    234    242
    256    260    262    265    267    270    277    285    289    297
    309    317    328    347    359    369    370    376    380    382
    386    399    404    406    408    412    414    430    438    441
    455    457    462    467    486    488    492    494    503    508
    521    530    539    547    548    551    555    558    559    563
    568    576    579    584    590    595    620    622    626    632
    636    649    651    653    657    663    669    672    679    681
    694    702    704    717    728    733    736    737    747    748
    756    761    762    780    785    789    793    801    806    810
```

812	813	816	819	835	836	841	842	845	847
848	850	857	862	875	877	887	895	902	903
907	910	925	930	932	935	938	944	955	956
962	964	966	968	970	973	980	987	989	1003
1004	1006	1016	1023	1030	1035	1036	1051	1052	1055
1060	1080	1083	1093	1109	1110	1116	1120	1122	1123
1133	1135	1138	1152	1160	1165	1167	1181	1185	1190
1195	1200	1201	1202	1204	1214	1222	1224	1226	1228
1233	1239	1240	1243	1244	1250	1253	1260	1261	1265
1268	1294	1307	1316	1322	1323	1324	1328	1334	1337
1339	1345	1349	1352	1355	1372	1390	1395	1401	1410
1414	1419	1428	1440	1441	1462	1465	1470	1475	1478
1482	1487	1490	1495	1500	1503	1513	1514	1520	1524
1527	1534	1536	1546	1547	1562	1569	1572	1575	1584
1597	1615	1618	1624	1626	1636	1642	1643	1644	1646
1673	1674	1683	1693	1701	1707	1709	1720	1722	1727
1730	1734	1742	1743	1746	1749	1765	1768	1769	1770
1772	1774	1775	1781	1785	1789	1796	1803	1804	1805
1806	1814	1823	1829	1841	1846	1848	1849	1862	1875
1886	1887	1888	1889	1894	1895	1897	1900	1903	1905
1906	1908	1911	1913	1914	1916	1917	1918	1921	1922
1929	1940	1947	1959	1964	1968	1972	1984	1989	1991
2005	2006	2016	2020	2025	2028	2042	2043	2050	2051
2052	2055	2060	2065	2086	2091	2103	2104	2112	2113
2118	2124	2127	2128	2132	2140	2146	2151	2164	2169
2171	2172	2175	2185	2187	2188	2192	2194	2203	2212
2230	2233	2234	2242	2246	2249	2256	2266	2267	2282
2283	2285	2299	2313	2316	2330	2334	2335	2336	2351
2352	2356	2357	2360	2367	2382	2386	2389	2392	2395
2396	2397	2405	2410	2422	2435	2436	2450	2482	2485
2490	2492	2493	2500	2502	2504	2508	2524	2525	2526
2527	2530	2534	2549	2557	2558	2564	2567	2580	2581
2595	2597	2601	2607	2610	2613	2615	2618	2621	2626
2637	2644	2647	2656	2665	2666	2678	2679	2685	2688
2689	2690	2691	2698	2712	2713	2717	2729	2739	2755
2758	2766	2770	2774	2782	2783	2784	2792	2793	2796
2799	2804	2807	2824	2829	2836	2837	2845	2847	2851
2853	2864	2868	2893	2899	2900	2903	2904	2916	2921
2924	2933	2934	2940	2945	2948	2957	2963	2971	2979
2981	2982	2987	2993	2994	3003	3005	3006	3009	3029
3032	3034	3035	3041	3044	3049	3053	3054	3056	3060
3064	3070	3078	3079	3084	3089	3090	3098	3107	3115
3118	3122	3128	3130	3137	3140	3143	3144	3145	3151
3153	3163	3165	3174	3177	3180	3186	3192	3197	3198
3206	3213	3225	3227	3228	3231	3240	3241	3244	3256
3258	3262	3263	3265	3271	3280	3282	3283	3299	3302
3306	3309	3335	3343	3350	3360	3361	3363	3365	3370
3371	3377	3379	3380	3381	3388	3392	3395	3396	3398
3426	3427	3430	3440	3442	3459	3466	3471	3479	3488
3489	3495	3506	3511	3519	3523	3525	3529	3531	3535
3544	3545	3546	3547	3552	3554	3561	3567	3571	3573
3575	3577	3578	3581	3584	3596	3602	3604	3608	3611

```
   3619   3626   3633   3636   3639   3647   3653   3655   3657   3658
   3668   3670   3677   3679   3680   3685   3690   3697   3698   3703
   3707   3711   3715   3726   3730   3750   3751   3756   3763   3770
   3771   3779   3781   3782   3787   3789   3794   3796   3799   3806
   3808   3811   3817   3820   3824   3825   3837   3842   3851   3860
   3872   3874   3876   3880   3894   3898   3899   3902   3907   3913
   3920   3926   3933   3945   3946   3949   3950   3954   3955   3958
   3969   3971   3979   3988   3992   3994   4000   4003   4006   4007
   4014   4018   4023   4026   4031   4042   4051   4072   4075   4076
   4079   4099   4100   4101   4104   4105   4118   4121   4145   4146
   4156   4166   4168   4173   4179   4181   4186   4191   4192   4196
   4200   4203   4211   4212   4213   4216   4221   4223   4225   4228
   4231   4246   4250   4252   4256   4258   4267   4269   4278   4288
   4301   4308   4313   4324   4331   4335   4349   4356   4363   4364
   4368   4372   4375   4378   4381   4383   4390   4391   4403   4407
   4408   4412   4415   4417   4420   4427   4429   4439   4441   4453
   4460   4462   4474   4476   4477   4480   4490   4491   4495   4513
   4520   4521   4529   4539   4542   4544   4546   4547   4565   4566
   4567   4570   4574   4578   4580   4597   4600   4604   4605   4622
   4632   4637   4642   4650   4664   4665   4671   4672   4691   4718
   4722   4726   4730   4739   4749   4767   4769   4773   4779   4785
   4786   4787   4788   4791   4794   4799   4815   4822   4823   4837
   4841   4847   4866   4886   4890   4891   4892   4902   4906   4912
   4914   4915   4930   4931   4932   4939   4942   4948   4950   4958
   4963   4973   4979   4988   4991   4993   4994   4995
Phase 10 (slow):
  Found 3840 at: -1 (is 0)
  Found 4995 at: 937 (is 4995)
  Found -24 at: 0 (is -24)
  Found 77 at: 40 (is 77)
  Found 4000 at: 766 (is 4000)
  Found 309 at: 90 (is 309)
  Found 2050 at: 398 (is 2050)
  Found 4539 at: 863 (is 4539)
Total search time was       26 milliseconds
Phase 10 (fast):
  Found 3840 at: -1 (is 0)
  Found 4995 at: 937 (is 4995)
  Found -24 at: 0 (is -24)
  Found 77 at: 40 (is 77)
  Found 4000 at: 766 (is 4000)
  Found 309 at: 90 (is 309)
  Found 2050 at: 398 (is 2050)
  Found 4539 at: 863 (is 4539)
Total search time was        3 milliseconds
%
```