**HOMEWORK 3**

YANG TANG ID: 53979886

1) Program output without valgrind (run on random.txt):

```
yangt8@andromeda-4 16:52:02 ~/hw/hw3
$ make
echo     ------------compiling main.ccp to create executable program main----------------
------------compiling main.ccp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
yangt8@andromeda-4 16:52:07 ~/hw/hw3
$ ./main
Testing SortedArrayList:
Inset all word: 6.84842
Find all word: 0.041519
Remove all word: 6.87071
----------------------------
Testing SortedLinkedList:
Inset all word: 15.2816
Find all word: 17.8925
Remove all word: 17.8676
```

2) Valgrind output (run on smaller test):

```
yangt8@andromeda-59 16:59:01 ~/hw/hw3
$ valgrind ./main
==23299== Memcheck, a memory error detector
==23299== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==23299== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==23299== Command: ./main
==23299==
Testing SortedArrayList:
Inset all word: 0.036673
Find all word: 0.006196
Remove all word: 0.005152
----------------------------
Testing SortedLinkedList:
Inset all word: 0.007668
Find all word: 0.004528
Remove all word: 0.006694
==23299==
==23299== HEAP SUMMARY:
==23299==     in use at exit: 0 bytes in 0 blocks
==23299==   total heap usage: 532 allocs, 532 frees, 430,423 bytes allocated
==23299==
==23299== All heap blocks were freed -- no leaks are possible
==23299==
==23299== For counts of detected and suppressed errors, rerun with: -v
==23299== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

3)

- SortedArrayList :

```
virtual void insert(string word) override   //O(N)
{
    int hole = binary_search(word);
    copy_down(hole);
    buf[hole] = word;
    size++;
}


int binary_search(string word)   //O(logN)
{
    int min = 0;
    int max = size-1;
    int mid;
    while (min <= max)
    {
            mid = (max-min)/2 + min;
            if (word < buf[mid])
            {
                    max = mid - 1;
            }
            else if (word > buf[mid])
                    min = mid + 1;
            else
                    return mid;
    }
    mid = (max-min)/2 + min;
    return mid;
}


void copy_down(int hole)   //O(N)
{
    for (int i = size; i > hole; i--)
            buf[i] = buf[i-1];
}


void insert_all_words(string file_name, SortedList & L)   //O(N^2)
{
     Timer t;
     double eTime;
     ifstream f(file_name);
     t.start();
     string w;
     while (f>>w)
     {
          L.insert(w);
```

```
    }
    f.close();
    t.elapsedUserTime(eTime);
    cout << "Inset all word: "<<eTime << endl;
}
```

- SortedLinkedList :

```
virtual void insert(string word) override   //O(N)
{
    if (head==nullptr || head->info > word)
        head = new ListNode{word,head};
    else
    {
        for (ListNode* p=head;p!=nullptr;p=p->next)
        {
            if (p->next != nullptr)
            {
                if (p->next->info > word)
                {
                    p->next = new ListNode{word,p->next};
                    break;
                }
            }
            else
            {
                if (p->info < word)
                    p->next = new ListNode{word,nullptr};
            }
        }
    }
}


void insert_all_words(string file_name, SortedList & L)   //O(N^2)
{
    Timer t;
    double eTime;
    ifstream f(file_name);
    t.start();
    string w;
    while (f>>w)
    {
        L.insert(w);
    }
    f.close();
    t.elapsedUserTime(eTime);
    cout << "Inset all word: "<<eTime << endl;
}
```