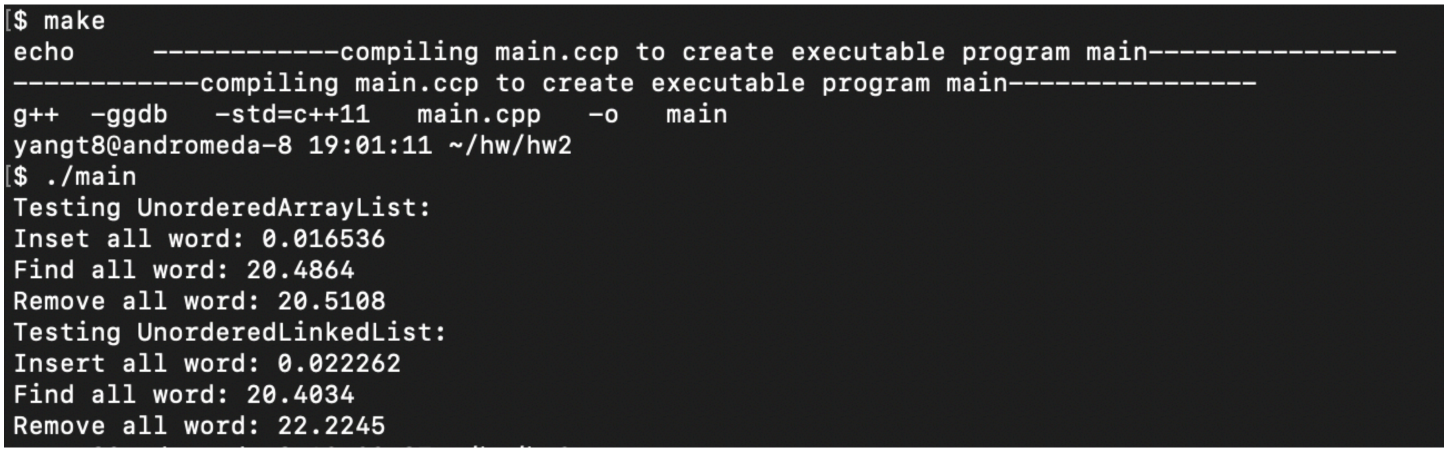
HOMEWORK 2

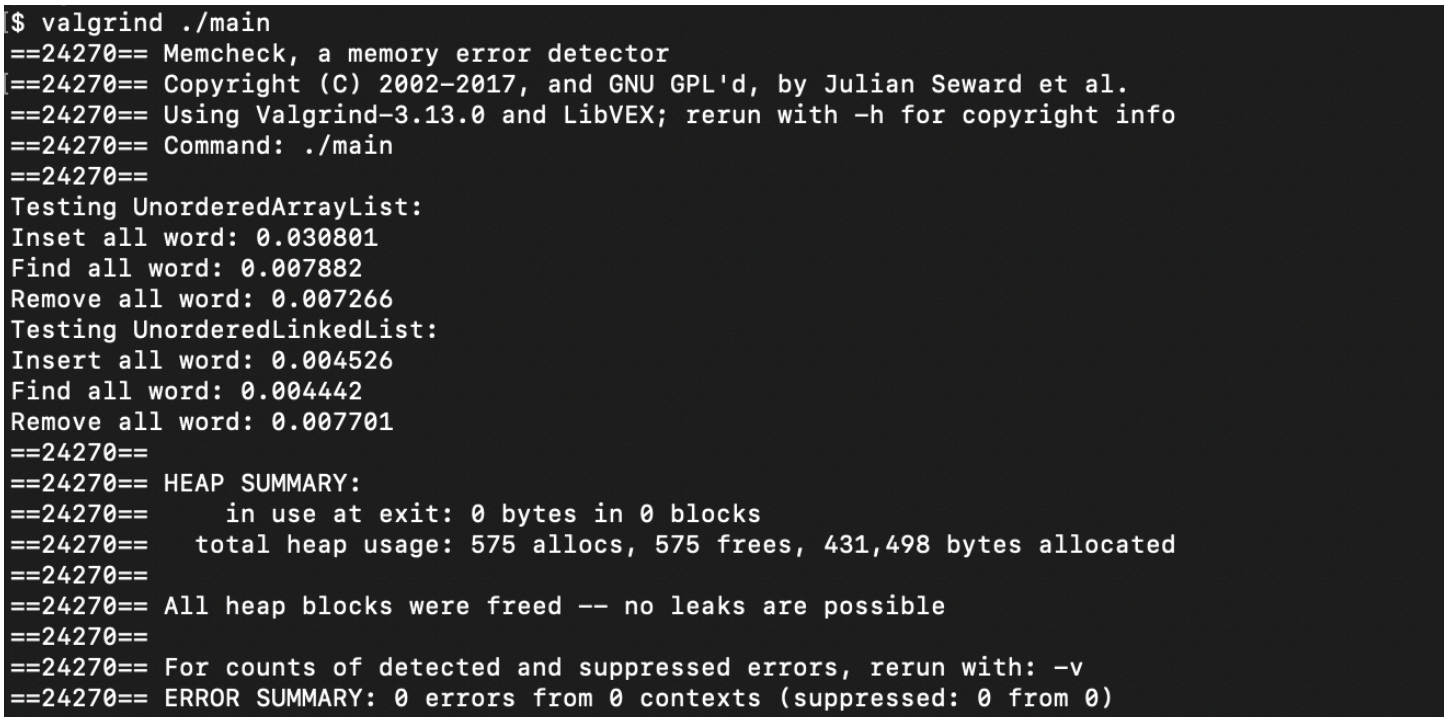
YANG TANG
ID: 53979886

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



```
[$ 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-8 19:01:11 ~/hw/hw2
[$ ./main
Testing UnorderedArrayList:
Inset all word: 0.016536
Find all word: 20.4864
Remove all word: 20.5108
Testing UnorderedLinkedList:
Insert all word: 0.022262
Find all word: 20.4034
Remove all word: 22.2245
```

2) Valgrind output (run on smaller test):



```
[$ valgrind ./main
==24270== Memcheck, a memory error detector
[==24270== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==24270== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==24270== Command: ./main
==24270==
Testing UnorderedArrayList:
Inset all word: 0.030801
Find all word: 0.007882
Remove all word: 0.007266
Testing UnorderedLinkedList:
Insert all word: 0.004526
Find all word: 0.004442
Remove all word: 0.007701
==24270==
==24270== HEAP SUMMARY:
==24270==     in use at exit: 0 bytes in 0 blocks
==24270==   total heap usage: 575 allocs, 575 frees, 431,498 bytes allocated
==24270==
==24270== All heap blocks were freed -- no leaks are possible
==24270==
==24270== For counts of detected and suppressed errors, rerun with: -v
==24270== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

3)

```
void UnorderedArrayList::insert(string word)  // O(1)
{
     if (isFull()==false)
          buf[size]=word;
          size++;
}


void UnorderedLinkedList::insert(string word)  // O(1)
{
        head = new ListNode{word,head};
}


void insert_all_words(string file_name, UnorderedArrayList & L) // O(N)
{
     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;
}



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



bool UnorderedArrayList::find(string word)  // O(N)
{
     for (int i =0;i<size;i++)
     {
          if (buf[i] == word)
               return true;
     }
     return false;
}


bool UnorderedLinkedList::find(string word)  // O(N)
{
        for (ListNode *p=head;p!=nullptr;p=p->next)
        {
                if (p->info == word)
                        return true;
        }
        return false;
}


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


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