HOMEWORK 4

YANG TANG ID: 53979886

1) Compilation

```
yangt8@andromeda-29 22:51:19 ~/hw/hw4
$ make
echo     ------------compiling test_stack.cpp and test_queue.cpp to create executable program main----------------
------------compiling test_stack.cpp and test_queue.cpp to create executable program main----------------
g++ -ggdb -std=c++0x -std=c++11 -Wpedantic -Wall -Wextra -Werror -Wzero-as-null-pointer-constant test_stack.cpp -o test_stack
g++ -ggdb -std=c++0x -std=c++11 -Wpedantic -Wall -Wextra -Werror -Wzero-as-null-pointer-constant test_queue.cpp -o test_queue
yangt8@andromeda-29 22:51:22 ~/hw/hw4
$ ./test_stack random.txt ArrayStackOutput.txt LinkedStackOutput.txt
---------TEST isBalanced---------
isBalanced('({(())})(((([{}]))))((((((<>([{()}])(<>))))))()''): 1
isBalanced('({(<>)})(((([{}]))))(((((((([{()}])(((((((''): 0
isBalanced('({(<>)})(((([{}]))))(((((((][([{<>}])(((((((()])''): 0
---------TEST Stack---------
yangt8@andromeda-29 22:52:09 ~/hw/hw4
$ ./test_queue random.txt ArrayQueueOutput.txt LinkedQueueOutput.txt
---------TEST Queue---------
```

2) Valgrind

```
yangt8@andromeda-29 22:52:25 ~/hw/hw4
$ valgrind ./test_stack random.txt ArrayStackOutput.txt LinkedStackOutput.txt
==12571== Memcheck, a memory error detector
==12571== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12571== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==12571== Command: ./test_stack random.txt ArrayStackOutput.txt LinkedStackOutput.txt
==12571==
---------TEST isBalanced---------
isBalanced('({(())})(((([{}]))))((((((<>([{()}])(<>))))))()''): 1
isBalanced('({(<>)})(((([{}]))))(((((((([{()}])(((((((''): 0
isBalanced('({(<>)})(((([{}]))))(((((((][([{<>}])(((((((()])''): 0
---------TEST Stack---------
==12571==
==12571== HEAP SUMMARY:
==12571==     in use at exit: 0 bytes in 0 blocks
==12571==   total heap usage: 227,206 allocs, 227,206 frees, 6,398,024 bytes allocated
==12571==
==12571== All heap blocks were freed -- no leaks are possible
==12571==
==12571== For counts of detected and suppressed errors, rerun with: -v
==12571== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
yangt8@andromeda-29 22:54:08 ~/hw/hw4
$ valgrind ./test_queue random.txt ArrayQueueOutput.txt LinkedQueueOutput.txt
==12596== Memcheck, a memory error detector
==12596== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12596== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==12596== Command: ./test_queue random.txt ArrayQueueOutput.txt LinkedQueueOutput.txt
==12596==
---------TEST Queue---------
==12596==
==12596== HEAP SUMMARY:
==12596==     in use at exit: 0 bytes in 0 blocks
==12596==   total heap usage: 227,058 allocs, 227,058 frees, 6,394,484 bytes allocated
==12596==
==12596== All heap blocks were freed -- no leaks are possible
==12596==
==12596== For counts of detected and suppressed errors, rerun with: -v
==12596== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

3) Time complexity

Linked Stack:

```
void fillAll(string file_name, Stack & L)   //O(N)
{
    try
    {
       ifstream f(file_name);
       if (f)
       {
          string w;
          while (f>>w)
          {
             L.push(w);
          }
          f.close();
       }
       else
          cout << "Error: Invalid file to read." << endl;
    }

    catch (ContainerOverflow& e)
    {
        cout << e.get_message()<< endl;
    }
}


void emptyAll(string file_name, Stack & L)   //O(N)
{
    try
    {
       ofstream f(file_name);
       if (f)
       {
          while (L.isEmpty() != true)
          {
             printout(f,L.pop());
          }
          f.close();
       }
       else
          cout << "Error: Invalid file to write." << endl;
    }
    catch (ContainerUnderflow& e)
    {
          cout << e.get_message()<< endl;
    }
}
```

```
void printout(ostream& out, string word)      //O(1)
{
    out << word << endl;
}

string get_message() const //O(1)
{
    return message;
}


virtual bool isEmpty() override   //O(1)
{
    return (head==nullptr);
}


virtual bool isFull() override   //O(1)
{
    return false;
}


virtual void push(string word) override   //O(1)
{
    if (isFull())
        throw ContainerOverflow("Error: Push on a full LinkedStack");
    head=new ListNode(word,head);
}


virtual string pop() override   //O(1)
{
    if (isEmpty())
        throw ContainerUnderflow("Error: Pop on an empty LinkedStack");
    ListNode* p = head;
    string o = head->info;
    head = head->next;
    delete p;
    return o;
}
```

Linked Queue:

```
void fillAll(string file_name, Queue & L)   //O(N)
{
    try
    {
       ifstream f(file_name);
       if (f)
       {
           string w;
           while (f>>w)
           {
              L.enq(w);
           }
           f.close();
        }
       else
           cout << "Error: Invalid file to read." << endl;
    }
    catch (ContainerOverflow& e)
    {
       cout << e.get_message()<< endl;
    }
}


void emptyAll(string file_name, Queue & L)   //O(N)
{
    try
    {
       ofstream f(file_name);
       if (f)
       {
          while (L.isEmpty() != true)
          {
             printout(f,L.deq());
           }
          f.close();
        }
       else
           cout << "Error: Invalid file to write." << endl;
    }
    catch (ContainerUnderflow& e)
    {
       cout << e.get_message()<< endl;
    }
}


void printout(ostream& out, string word)   //O(1)
{
    out << word << endl;
```

```
}


string get_message() const //O(1)
{
    return message;
}


virtual bool isEmpty() override   //O(1)
{
    return (head == nullptr);
}


virtual bool isFull() override   //O(1)
{
    return false;
}


virtual void enq(string word) override    //O(1)
{
    ListNode* newNode = new ListNode(word,nullptr);
    if (isEmpty())
          head = newNode;
    else
          tail->next = newNode;
    tail = newNode;
}


virtual string deq() override   //O(1)
{
    if (isEmpty())
        throw ContainerUnderflow("Error: Deque on an empty LinkedQueue");
    ListNode* p = head;
    string o = head->info;
    head = head->next;
    delete p;
    return o;
}
```