Name: Yu Qin

Results Screenshots

==========start of the write-up==========

Lab 1

```
$ pwd
/home/yuq8/253P/hw_lab/4
yuq8@andromeda-30 13:12:25 ~/253P/hw_lab/4
$ ls
.DS_Store             HW_Report_yuq8.docx  Lab4_Report_yuq8.docx  MCS 253p HW4.docx
hw1_process_numbers/  lab1_product/        Lab4_Report_yuq8.pdf   ~$_Report_yuq8.docx
hw2_freq_stop/        lab2_reversePolish/  MCS 253 Lab 4.docx
yuq8@andromeda-30 13:12:29 ~/253P/hw_lab/4
$ cd lab1_product/
yuq8@andromeda-30 13:12:43 ~/253P/hw_lab/4/lab1_product
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main
yuq8@andromeda-30 13:12:54 ~/253P/hw_lab/4/lab1_product
$
```

```
$ valgrind ./main
==24942== Memcheck, a memory error detector
==24942== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==24942== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==24942== Command: ./main
==24942==
input the number of integers
5
input integers
10
3
5
6
2
180
600
360
300
900
==24942==
==24942== HEAP SUMMARY:
==24942==     in use at exit: 72,704 bytes in 1 blocks
==24942==   total heap usage: 1 allocs, 0 frees, 72,704 bytes allocated
==24942==
==24942== LEAK SUMMARY:
==24942==    definitely lost: 0 bytes in 0 blocks
==24942==    indirectly lost: 0 bytes in 0 blocks
==24942==      possibly lost: 0 bytes in 0 blocks
==24942==    still reachable: 72,704 bytes in 1 blocks
==24942==         suppressed: 0 bytes in 0 blocks
==24942== Rerun with --leak-check=full to see details of leaked memory
==24942==
==24942== For counts of detected and suppressed errors, rerun with: -v
==24942== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 13:14:46 ~/253P/hw_lab/4/lab1_product
$
```

```
$ valgrind ./main
==25047== Memcheck, a memory error detector
==25047== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==25047== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==25047== Command: ./main
==25047==
input the number of integers
4
input integers
-1
1
5
10
50
-50
-10
-5
==25047==
==25047== HEAP SUMMARY:
==25047==     in use at exit: 72,704 bytes in 1 blocks
==25047==   total heap usage: 1 allocs, 0 frees, 72,704 bytes allocated
==25047==
==25047== LEAK SUMMARY:
==25047==    definitely lost: 0 bytes in 0 blocks
==25047==    indirectly lost: 0 bytes in 0 blocks
==25047==      possibly lost: 0 bytes in 0 blocks
==25047==    still reachable: 72,704 bytes in 1 blocks
==25047==         suppressed: 0 bytes in 0 blocks
==25047== Rerun with --leak-check=full to see details of leaked memory
==25047==
==25047== For counts of detected and suppressed errors, rerun with: -v
==25047== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 13:15:27 ~/253P/hw_lab/4/lab1_product
$
```

Conclusion: right result, 0 error in memory leak.

Lab 2

```
$ cd lab2_reversePolish/
yuq8@andromeda-30 13:16:29 ~/253P/hw_lab/4/lab2_reversePolish
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main
```

```
$ valgrind ./main input
==25519== Memcheck, a memory error detector
==25519== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==25519== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==25519== Command: ./main input
==25519==
256
-2222221824
64
success!
==25519==
==25519== HEAP SUMMARY:
==25519==     in use at exit: 72,704 bytes in 1 blocks
==25519==   total heap usage: 31 allocs, 30 frees, 87,476 bytes allocated
==25519==
==25519== LEAK SUMMARY:
==25519==    definitely lost: 0 bytes in 0 blocks
==25519==    indirectly lost: 0 bytes in 0 blocks
==25519==      possibly lost: 0 bytes in 0 blocks
==25519==    still reachable: 72,704 bytes in 1 blocks
==25519==         suppressed: 0 bytes in 0 blocks
==25519== Rerun with --leak-check=full to see details of leaked memory
==25519==
==25519== For counts of detected and suppressed errors, rerun with: -v
==25519== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 13:17:31 ~/253P/hw_lab/4/lab2_reversePolish
$
```

Conclusion: right result, 0 error in memory leak.

Little Notes:

1) Future work can be taken in transforming the prefix notation to RPN, plus the question of taking parenthesis into consideration.

HW1

```
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main
yuq8@andromeda-30 13:25:19 ~/253P/hw_lab/4/hw1_process_numbers
$ valgrind ./main
==27504== Memcheck, a memory error detector
==27504== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==27504== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==27504== Command: ./main
==27504==
successed in process and save!
==27504==
==27504== HEAP SUMMARY:
==27504==     in use at exit: 72,704 bytes in 1 blocks
==27504==   total heap usage: 16 allocs, 15 frees, 101,028 bytes allocated
==27504==
==27504== LEAK SUMMARY:
==27504==    definitely lost: 0 bytes in 0 blocks
==27504==    indirectly lost: 0 bytes in 0 blocks
==27504==      possibly lost: 0 bytes in 0 blocks
==27504==    still reachable: 72,704 bytes in 1 blocks
==27504==         suppressed: 0 bytes in 0 blocks
==27504== Rerun with --leak-check=full to see details of leaked memory
==27504==
==27504== For counts of detected and suppressed errors, rerun with: -v
==27504== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 13:25:26 ~/253P/hw_lab/4/hw1_process_numbers
$ cat even.txt
-98     -96     -94     -92     -90     -88     -86     -84     -82     -80     -78     -76     -74     -72     -70     -68     -66     -64     -62     -60     -58     -56     -54 -
52      -50     -48     -46     -44     -42     -40     -38     -36     -34     -32     -30     -28     -26     -24     -22     -20     -18     -16     -14     -12     -10     -8  -
6       -4      -2      0       2       4       6       8       10      12      14      16      18      20      22      24      26      28      30      32      34      36      38  4
0       42      44      46      48      50      52      54      56      58      60      62      64      66      68      70      72      74      76      78      80      82      84  8
6       88      90      92      94      96      98      100     yuq8@andromeda-30 13:25:37 ~/253P/hw_lab/4/hw1_process_numbers
$ cat odd.txt
-99     -97     -95     -93     -91     -89     -87     -85     -83     -81     -79     -77     -75     -73     -71     -69     -67     -65     -63     -61     -59     -57     -55 -
53      -51     -49     -47     -45     -43     -41     -39     -37     -35     -33     -31     -29     -27     -25     -23     -21     -19     -17     -15     -13     -11     -9  -
7       -5      -3      -1      1       3       5       7       9       11      13      15      17      19      21      23      25      27      29      31      33      35      37  3
9       41      43      45      47      49      51      53      55      57      59      61      63      65      67      69      71      73      75      77      79      81      83  8
5       87      89      91      93      95      97      99      yuq8@andromeda-30 13:25:43 ~/253P/hw_lab/4/hw1_process_numbers
$
```

Conclusion: right result, 0 error in memory leak.
Little Notes:
1) Interesting and useful of using complex c++ STL.