Test Plan and Test Log for Machine Problem 2

Student: Yulin Xiao

CUID: C16278133

Apollo User Name: Yulinx(yulinx)

ABSTRACT:

For the Machine Problem 2, there are a lot of operations that are available to test the double linked list in Abstract Data Representations. Here are my test plans:

Test Plan:

1. First: modify the inputs and outputs from the Instructor given files.
2. Second: Test the boundaries

1) fill the tasks
2) modify the tasks in the zero or one element
3) modify the tasks in the unexpected input or output.
4) check the sorted array after REVERSE.

Test Log:

1. input1[1]:

```
INSERT
1
1
1
1
INSERT
2
2
2
2
2
INSERT
3
3
3
3
3
3
STATS
PRINT
INSERT
1
```

1
1
1
INSERT
2
2
2
2
2
INSERT
3
3
3
3
3
3
STATS
PRINT
FIND 1
FIND 2
FIND 3
REMOVE 1
REMOVE 2
REMOVE 3
QUIT

Output1[0]:
Lab2 list size is 10. Possible commands:
List: INSERT
FIND id
REMOVE id
UPDATE id state
SCHEDULE id priority
DETERMINE
REVERSE
PRINT
Queue : ADDTAIL; RMHEAD; PRINTHEAD; PRINTQ
: STATS; QUIT
Priority:Wallclocktime:Number of Args:Arg 0:
Inserted: 0
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:
Inserted: 1
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:Arg 2:
Inserted: 2

```
Number records: 3, Order: Ascending
Number records in queue: 0
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 1 priority = 2 state = QUEUED
    2: Task ID: 2 priority = 3 state = QUEUED
Priority:Wallclocktime:Number of Args:Arg 0:
Inserted: 3
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:
Inserted: 4
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:Arg 2:
Inserted: 5
Number records: 6, Order: Ascending
Number records in queue: 0
List has 6 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 3 priority = 1 state = QUEUED
    2: Task ID: 1 priority = 2 state = QUEUED
    3: Task ID: 4 priority = 2 state = QUEUED
    4: Task ID: 2 priority = 3 state = QUEUED
    5: Task ID: 5 priority = 3 state = QUEUED
Found: 1 at index 2
Task ID: 1
        priority = 2
        state        = QUEUED
        time = 2.000000e+00
        nargs = 2
        args = { 2,    2, }
Found: 2 at index 4
Task ID: 2
        priority = 3
        state        = QUEUED
        time = 3.000000e+00
        nargs = 3
        args = { 3,    3,    3, }
Found: 3 at index 1
Task ID: 3
        priority = 1
        state        = QUEUED
        time = 1.000000e+00
        nargs = 1
        args = { 1, }
Removed: 1
Task ID: 1
```

priority = 2
state = QUEUED
time = 2.000000e+00
nargs = 2
args = { 2, 2, }
Removed: 2
Task ID: 2
priority = 3
state = QUEUED
time = 3.000000e+00
nargs = 3
args = { 3, 3, 3, }
Removed: 3
Task ID: 3
priority = 1
state = QUEUED
time = 1.000000e+00
nargs = 1
args = { 1, }
Goodbye
Reason Why I test like this: Normal Operations are correct

2. input2[1]:

INSERT
1
1
1
1
INSERT
2
2
2
2
2
INSERT
3
3
3
3
3
3
REVERSE
STATS
PRINT
INSERT

1
1
1
1
INSERT
3
3
3
3
3
3
INSERT
2
2
2
2
2
STATS
PRINT
REVERSE
STATS
PRINT
QUIT

Output2[1]:
Lab2 list size is 10. Possible commands:
List: INSERT
FIND id
REMOVE id
UPDATE id state
SCHEDULE id priority
DETERMINE
REVERSE
PRINT
Queue : ADDTAIL; RMHEAD; PRINTHEAD; PRINTQ
: STATS; QUIT
Priority:Wallclocktime:Number of Args:Arg 0:
Inserted: 0
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:
Inserted: 1
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:Arg 2:
Inserted: 2
List reversed, new order: Descending
Number records: 3, Order: Descending

Number records in queue: 0
List has 3 records
    0: Task ID: 2 priority = 3 state = QUEUED
    1: Task ID: 1 priority = 2 state = QUEUED
    2: Task ID: 0 priority = 1 state = QUEUED
Priority:Wallclocktime:Number of Args:Arg 0:
Inserted: 3
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:Arg 2:
Inserted: 4
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:
Inserted: 5
Number records: 6, Order: Descending
Number records in queue: 0
List has 6 records
    0: Task ID: 4 priority = 3 state = QUEUED
    1: Task ID: 2 priority = 3 state = QUEUED
    2: Task ID: 5 priority = 2 state = QUEUED
    3: Task ID: 1 priority = 2 state = QUEUED
    4: Task ID: 3 priority = 1 state = QUEUED
    5: Task ID: 0 priority = 1 state = QUEUED
List reversed, new order: Ascending
Number records: 6, Order: Ascending
Number records in queue: 0
List has 6 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 3 priority = 1 state = QUEUED
    2: Task ID: 1 priority = 2 state = QUEUED
    3: Task ID: 5 priority = 2 state = QUEUED
    4: Task ID: 2 priority = 3 state = QUEUED
    5: Task ID: 4 priority = 3 state = QUEUED
Goodbye
Reason Why I test like this: Try to see the order after reverse and the insert position will not change after reverse: Always insert after the priority is the same. That's how I interpret the meaning of this: *if the element to be inserted is equal in rank to an element already in the list, the newly inserted element will be placed after all the elements of equal rank that are already in the list.*

3. Input3[1]:

DETERMINE

1
1
DETERMINE
2
2
2
FIND 1
FIND 2
FIND 22
FIND -1
REMOVE 1
REMOVE 2
REMOVE 2
REMOVE -1
UPDATE 1 3
UPDATE 1 2
UPDATE 1 1
STATS
PRINT
SCHEDULE 1 1
SCHEDULE 2 2
SCHEDULE 3 3
CLEAN
REVERSE
STATS
PRINT
REVERSE
STATS
PRINT
RMHEAD
PRINTHEAD
PRINTQ
QUIT

Output3[0]:
Lab2 list size is 10. Possible commands:
List: INSERT
FIND id
REMOVE id
UPDATE id state
SCHEDULE id priority
DETERMINE
REVERSE
PRINT

Queue : ADDTAIL; RMHEAD; PRINTHEAD; PRINTQ
: STATS; QUIT
Number of Args:Arg 0:No runable tasks.
Number of Args:Arg 0:Arg 1:No runable tasks.
Did not find: 1
Did not find: 2
Did not find: 22
Did not find: -1
Did not remove: 1
Did not remove: 2
Did not remove: 2
Did not remove: -1
Number records: 0, Order: Ascending
Number records in queue: 0
List empty
Did not find task to schedule...
Did not find task to schedule...
Did not find task to schedule...
Removing 0 finished tasks
List reversed, new order: Descending
Number records: 0, Order: Descending
Number records in queue: 0
List empty
List reversed, new order: Ascending
Number records: 0, Order: Ascending
Number records in queue: 0
List empty
Queue empty, did not remove
Queue is empty
Queue empty
Goodbye

Why I test like this: The unexpected delete or find and unexpected for queue list.

4. Input4[1].txt
INSERT
1
1
1
1
INSERT
2
2
2

2
2
INSERT
3
3
3
3
3
3
STATS
PRINT
DETERMINE
1
1
DETERMINE
2
2
2
DETERMINE
3
3
3
3
UPDATE 2 1
PRINT
UPDATE 2 2
PRINT
UPDATE 2 3
DETERMINE
3
3
3
3
DETERMINE
3
3
3
3
UPDATE 1 1
PRINT
UPDATE 1 2
PRINT
UPDATE 1 3
PRINT

DETERMINE
2
2
2
STATS
PRINT
CLEAN
STATS
PRINT
DETERMINE
1
1
DETERMINE
1
1
UPDATE 0 1
PRINT
UPDATE 0 2
PRINT
UPDATE 0 3
PRINT
DETERMINE
1
1
STATS
PRINT
CLEAN
STATS
PRINT
REMOVE 0
REMOVE 1
REMOVE 2
REMOVE -1
QUIT

Output4[0]:
Lab2 list size is 10. Possible commands:
List: INSERT
FIND id
REMOVE id
UPDATE id state
SCHEDULE id priority
DETERMINE
REVERSE

```
PRINT
Queue           : ADDTAIL; RMHEAD; PRINTHEAD; PRINTQ
                : STATS; QUIT
Priority:Wallclocktime:Number of Args:Arg 0:
Inserted: 0
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:
Inserted: 1
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:Arg 2:
Inserted: 2
Number records: 3, Order: Ascending
Number records in queue: 0
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 1 priority = 2 state = QUEUED
    2: Task ID: 2 priority = 3 state = QUEUED
Number of Args:Arg 0:Task 0 is runable.
Number of Args:Arg 0:Arg 1:Task 1 is runable.
Number of Args:Arg 0:Arg 1:Arg 2:Task 2 is runable.
Task 2 has state of RUNNING
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 1 priority = 2 state = QUEUED
    2: Task ID: 2 priority = 3 state = RUNNING
Task 2 has state of BLOCKED
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 1 priority = 2 state = QUEUED
    2: Task ID: 2 priority = 3 state = BLOCKED
Task 2 has state of FINISHED
Number of Args:Arg 0:Arg 1:Arg 2:No runable tasks.
Number of Args:Arg 0:Arg 1:Arg 2:No runable tasks.
Task 1 has state of RUNNING
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 1 priority = 2 state = RUNNING
    2: Task ID: 2 priority = 3 state = FINISHED
Task 1 has state of BLOCKED
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 1 priority = 2 state = BLOCKED
    2: Task ID: 2 priority = 3 state = FINISHED
Task 1 has state of FINISHED
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
```

```
    1: Task ID: 1 priority = 2 state = FINISHED
    2: Task ID: 2 priority = 3 state = FINISHED
Number of Args:Arg 0:Arg 1:No runable tasks.
Number records: 3, Order: Ascending
Number records in queue: 0
List has 3 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 1 priority = 2 state = FINISHED
    2: Task ID: 2 priority = 3 state = FINISHED
Removing 1 finshed tasks
Task ID: 1 priority = 2 state = FINISHED
Number records: 2, Order: Ascending
Number records in queue: 0
List has 2 records
    0: Task ID: 0 priority = 1 state = QUEUED
    1: Task ID: 2 priority = 3 state = FINISHED
Number of Args:Arg 0:Task 0 is runable.
Number of Args:Arg 0:Task 0 is runable.
Task 0 has state of RUNNING
List has 2 records
    0: Task ID: 0 priority = 1 state = RUNNING
    1: Task ID: 2 priority = 3 state = FINISHED
Task 0 has state of BLOCKED
List has 2 records
    0: Task ID: 0 priority = 1 state = BLOCKED
    1: Task ID: 2 priority = 3 state = FINISHED
Task 0 has state of FINISHED
List has 2 records
    0: Task ID: 0 priority = 1 state = FINISHED
    1: Task ID: 2 priority = 3 state = FINISHED
Number of Args:Arg 0:No runable tasks.
Number records: 2, Order: Ascending
Number records in queue: 0
List has 2 records
    0: Task ID: 0 priority = 1 state = FINISHED
    1: Task ID: 2 priority = 3 state = FINISHED
Removing 1 finshed tasks
Task ID: 0 priority = 1 state = FINISHED
Number records: 1, Order: Ascending
Number records in queue: 0
List has 1 record.
    0: Task ID: 2 priority = 3 state = FINISHED
Did not remove: 0
Did not remove: 1
```

Removed: 2
Task ID: 2
priority = 3
state = FINISHED
time = 3.000000e+00
nargs = 3
args = { 3, 3, 3, }
Did not remove: -1
Goodbye
Why I test like this: Try to SCHEDULE, DETERMINE and REMOVE FINISHED and Boundary
5.Input5[1]:
INSERT
1
1
1
1
INSERT
2
2
2
2
2
INSERT
3
3
3
3
3
3
STATS
PRINT
ADDTAIL
1
1
1
1
STATS
PRINTQ
RMHEAD
ADDTAIL
1
1
1

1
ADDTAIL
2
2
2
2
2
PRINTHEAD
RMHEAD
STATS
PRINTQ
PRINT
QUIT
Output5[0]:
Lab2 list size is 10. Possible commands:
List: INSERT
FIND id
REMOVE id
UPDATE id state
SCHEDULE id priority
DETERMINE
REVERSE
PRINT
Queue : ADDTAIL; RMHEAD; PRINTHEAD; PRINTQ
: STATS; QUIT
Priority:Wallclocktime:Number of Args:Arg 0:
Inserted: 0
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:
Inserted: 1
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:Arg 2:
Inserted: 2
Number records: 3, Order: Ascending
Number records in queue: 0
List has 3 records
0: Task ID: 0 priority = 1 state = QUEUED
1: Task ID: 1 priority = 2 state = QUEUED
2: Task ID: 2 priority = 3 state = QUEUED
Priority:Wallclocktime:Number of Args:Arg 0:
Appended 3 onto queue
Number records: 3, Order: Ascending
Number records in queue: 1
Queue has 1 record.
0: Task ID: 3 priority = 1 state = QUEUED
Deleted head with task id and priority: 3 1

Priority:Wallclocktime:Number of Args:Arg 0:
Appended 4 onto queue
Priority:Wallclocktime:Number of Args:Arg 0:Arg 1:
Appended 5 onto queue
Found at head of queue:
Task ID: 4
priority = 1
state = QUEUED
time = 1.000000e+00
nargs = 1
args = { 1, }
Deleted head with task id and priority: 4 1
Number records: 3, Order: Ascending
Number records in queue: 1
Queue has 1 record.
0: Task ID: 5 priority = 2 state = QUEUED
List has 3 records
0: Task ID: 0 priority = 1 state = QUEUED
1: Task ID: 1 priority = 2 state = QUEUED
2: Task ID: 2 priority = 3 state = QUEUED
Goodbye
Why I want to test this: test if there is an memory leak. But because I can not change the fill_task_ptr(), so that the boundary test failed when INSERT but do not type in any thing!