King Abdulaziz University
Faculty of Engineering
Electrical & Computer Eng
Department

LAB #4

Operating Systems– EE463

Name/ Saleh Ali Khalaf #1637360

1)The memory leak issue lies within the remove() function. It appears that when an element is removed from the beginning of the list, the head_ pointer is reassigned to a new node, but the previous node is not properly deleted. A simple solution to address this problem is by including a delete statement. Consider the following modified code:

In the case where marker->next() equals 0:

if (marker->next() == 0) {

```
head_ = 0;
delete marker;  // Solution: Explicitly delete the old node
marker = 0;
}
```

Alternatively, if marker->next() is not 0:

```
else {
head_ = new Node(marker->value(), marker->next());
delete marker;  // Solution: Explicitly delete the old node
marker = 0;
}
```

By adding the delete statement, we ensure that the old node is properly removed from memory, resolving the memory leak issue.

2) The other issue that we've noticed is that when we attempt to remove an element, the 'next' pointer for the preceding node fails to update. To address this, we can introduce an additional line: temp->next(marker->next()); This adjustment should be made both for instances when we are eliminating from the list's starting position as well as from the middle. Here's how we can integrate this:

If the 'next' pointer of our marker node points to null (i.e., marker->next() == 0), we adjust the head_ to null, eliminate the marker, and set marker to null. The altered code is as follows:

```
if (marker->next() == 0) {
head_ = 0;
delete marker;
marker = 0;
```

```
}
```

Otherwise, we initialize a new node for the head_, give it the value of the marker, and the 'next' pointer of the marker. Post that, we delete the marker and then reset the 'next' pointer of the temp node to the 'next' pointer of the marker. Now, we set marker to null. The corresponding code changes can be seen as:

```
else {
head_ = new Node(marker->value(), marker->next());
delete marker;
temp->next(marker->next());
marker = 0;
}
```

We then have to delete the temp node and, crucially, adjust the 'next' pointer of the temp node to the 'next' pointer of the marker node, a line we just added. After this, we set temp to null. The final part of our code will look like this:

```
temp->next(marker->next());
delete temp;
temp->next(marker->next());  // NEW LINE ADDED
temp = 0;
```