HW W3 Complexity Table

For each of the functions you wrote, specify the order of complexity and explain it. For example, if you specify that the complexity was $o(n)$, you should explain that you used a *for* loop to transverse the linked list for the task you needed.

1) *Add Student* Screenshot

```
StudentNode* newNode = new StudentNode();
newNode->setID(student_ID);
newNode->setName(student_name);
if(this->first == nullptr)
{
   this->first = newNode;
}
else
{
   newNode->setNext(this->first);
   this->first = newNode;
}
this->length++;
```

2) *Add Grade* Screenshot

```
StudentNode* current_location = first;
bool found = false;

while(found == false)
{
   if(current_location->getID() == student_ID)
   {
      current_location->setHW1(hw1_score);
      found = true;
   }
   else
```

```
   {
      current_location = current_location->getNext();
   }
}
```

3) *Delete Student* Screenshot

```
StudentNode* current_location = first;
StudentNode* temp_location;

if(first->getID() == student_ID)
{
   temp_location = current_location;
   first = first->getNext();
}
else
{
   while(current_location->getNext()->getID() != student_ID)
   {
      current_location = current_location->getNext();
   }
   temp_location = current_location->getNext();
   current_location->setNext(current_location->getNext()->getNext());
}
delete temp_location;
length--;
```

4) *Change Grade* Screenshot

```
      int current = my_list.first->getID();
      StudentNode* next = my_list.first;

      while(current != student_ID)
```

```
        {
            next = next->getNext();
            current = next->getID();
        }
        next->setHW1(grade);
```

5) *Print Student* Screenshot

```
        int current = my_list.first->getID();
        StudentNode* next = my_list.first->getNext();
        while(current != student_ID)
        {
            current = next->getID();
            if(current == student_ID)
            {
                next->printMe();
            }
            next = next->getNext();
        }
```

6) *Print Assignment* Screenshot

```
if(homework == "HW1")
        {
            total_score += my_list.first->getHW1();
            StudentNode* next = my_list.first->getNext();
            while(next != nullptr)
            {
                total_score += next->getHW1();
                next = next->getNext();
            }
            hw1_mean = total_score/length;
        }
```

7) *Print Top Student* Screenshot

```
student = my_list.first->getName();
StudentNode* next = my_list.first->getNext();
while(next != nullptr)
{

   if(next->getHW1() > high_score)
   {
      high_score = next->getHW1();
      student = next->getName();
   }
   next = next->getNext();
}
```

8) *Print* Screenshot

```
while(next != nullptr)
 {
    next = next->getNext();
 }
```

Command	Complexity	Explanation
ADD_STUDENT	O(1)	The code just adds a node to the front of the list.
ADD_GRADE	O(n)	The code iterates over the nodes until it finds the matching ID number.
DELETE_STUDENT	O(n)	The code iterates over the nodes until it finds the matching ID number.
CHANGE_GRADE	O(n)	The code iterates over the nodes until it finds the matching ID number.
PRINT_STUDENT	O(n)	The code iterates over the nodes until it finds the matching ID number.
PRINT_ASSIGNMENT	O(n)	The code iterates over the nodes to collect info on a particular assignment.
PRINT_TOP_STUDENT		The code iterates over the nodes and saves the

		highest grade for an assignment.
PRINT	O(n)	The code iterates over the nodes until it finds the matching ID number.
QUIT	O(1)	The code changes bool quit from false to true.