

```

bool deleteNode(int index)
{
    if (index < 0 || index >= count)           c1 = 1
        return false;                          c2 = 1
    if (index == 0)                             c3 = 1
    {
        return pop_front();                    c4 = 1
    }
    else if (index == count - 1)                c5 = 1
    {
        return pop_back();                     c7 = 1
    }
    else
    {
        if(index <= count / 2) // if closer from head   c8 = 1
        {
            Node<T>* current = head;                c9 = 1
            for (int i = 0; i < index; ++i)           c10 = 1 , c11 = n/2
            {
                current = current->next;              c12 = n/2
            }                                         c13(i++) = n/2
            return deleteAfter(current->prev);        c14 = 1
        }
        else // if closer from tail
        {
            Node<T>* current = tail;                 c145 = 1
            for (int i = count - 1; i > index; --i)    c16 = , C17 = n/2
            {
                current = current->prev;              c18 = n/2
            }                                         c19(i++) = n/2
            return deleteAfter(current->prev);        c20 = 1
        }
    }
}

```

Cost = C1 + C20

$$T(n) = 1 + 1 + + n/2 + n/2 + n/2 + n/2 + n/2 + n/2 + 1$$

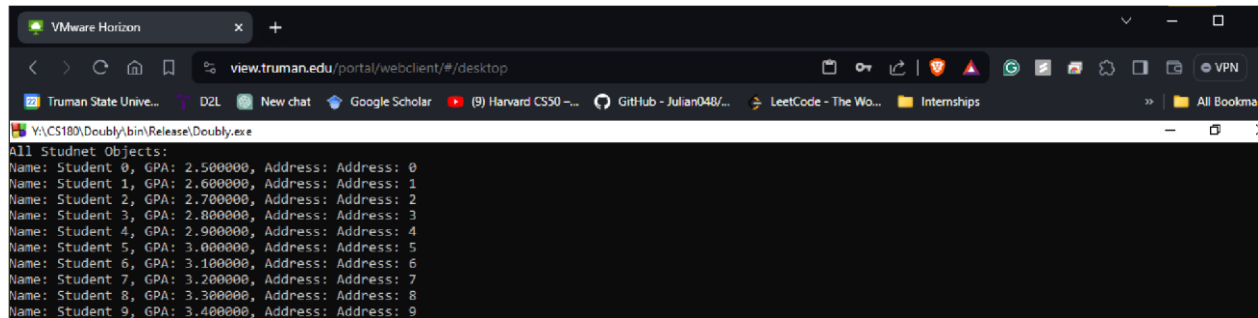
$$3n + 11 \leq 10n$$

For any values $n \geq 10$

Thus, $f(n) = n$, So $O(f(n)) = O(n)$.

This algorithm has the big O of n , in the worst case it will perform linear, but most of the time the time complexity is $n/2$ as it is iterating through half of the element.

Deliverable 8 Program Screenshot



The screenshot shows a VMware Horizon desktop environment. At the top, there is a browser window with the address bar showing 'view.truman.edu/portal/webclient/#/desktop'. Below the browser, there is a terminal window titled 'Y:\CS100\Doubly\bin\Release\Doubly.exe'. The terminal displays the output of a program, listing all student objects with their names, GPAs, and addresses.

```
All Student Objects:  
Name: Student 0, GPA: 2.500000, Address: Address: 0  
Name: Student 1, GPA: 2.600000, Address: Address: 1  
Name: Student 2, GPA: 2.700000, Address: Address: 2  
Name: Student 3, GPA: 2.800000, Address: Address: 3  
Name: Student 4, GPA: 2.900000, Address: Address: 4  
Name: Student 5, GPA: 3.000000, Address: Address: 5  
Name: Student 6, GPA: 3.100000, Address: Address: 6  
Name: Student 7, GPA: 3.200000, Address: Address: 7  
Name: Student 8, GPA: 3.300000, Address: Address: 8  
Name: Student 9, GPA: 3.400000, Address: Address: 9
```