Problem 3 Analysis

SimpleVector Push

=> T * naptr;

```
try {
  naptr = new T[arraySize+1];
} catch (bad_alloc) {
    memError();
}
for(int count=0; count<arraySize; count++) {
    naptr[count] = aptr[count];
}
naptr[arraySize++] = val;
delete [] aptr;
aptr = 0;
aptr = naptr;
```

O_b (try/catch block)

O_a (remaining statements)

$$\Rightarrow O_b + \sum_{i=0}^{n-1} O_i + O_n$$

$$\Rightarrow (O_b + O_a) + (N-1+1)\,O_i$$

$$\Rightarrow (O_b + O_a) + O_i N$$

$$\Rightarrow C_0 + C_1 N$$

$$\Rightarrow \boxed{f(n) = O(n)}$$ ← on average

Simple Vector Linked List

```
void LinkedList<T>::addList(const T& data){
    Link* end;
    temp=front;
    do{
      end=temp;
      temp=temp->linkPtr;
    }while(temp != Null);
    Link* add = new Link;
    add->data=data;
    add->LinkPtr=Null;
    end->LinkPtr=add;
}
```

$$\Rightarrow O_b + \sum_{i=0}^{n-1} O_i + O_a$$

$$\Rightarrow O_b + O_a + (N-1+1)O_i$$

$$\Rightarrow (O_b + O_a) + O_i(N)$$

$$\Rightarrow C_0 + C_1 N = f(N)$$

$$\Rightarrow \boxed{f(N) = O(N)}$$ ← on average

Optimized SimpleVector Push

```
if (arraySize == maxSize) {
    maxSize *= 2;
    T * naptr;
    try {
      naptr = new T[maxSize];
    }
    catch (bad_alloc) {
        memError();
    }
    for (int count=0; count<arraySize; count++) {
        naptr[count] = aptr[count];
    }
    naptr[arraySize++] = val;
    delete [] aptr;
    aptr=0;
    aptr=naptr;
}
else {
    aptr[arraySize++] = val;
}
```

on average

$F(N) = O(N)$ ~~on average~~