Problem 1 (20 points)

Answer:

1. *default constructor*
2. *conversion constructor*
3. *assignment operator*
4. *copy constructor*
5. *(5, 0, 0)*
6. *conversion constructor*
7. *In destructor*
8. *In destructor*
9. *In destructor*
10. *In destructor*

Marking Scheme: 10 blanks, 2 points each, 20 points in total.
NOTE: For the 5th blank, if you write “5, 0, 0” (or “5 0 0”, etc.), you will be deducted 1 point.

Problem 2 (50 points)

1.

```
Complex::Complex(double real, double imaginary)
{
    this->real = real;
    this->imaginary = imaginary;
}
```

2.

```
c = 5 + 0i
```

3.

```
Complex Complex::operator*(const Complex& c) const
{
    return Complex(real * c.real - imaginary * c.imaginary, real * c.imaginary + imaginary *
c.real);
}
```

4.

```
ostream& operator<<(ostream& out, const Complex c)
{
    static int counter;
    out << "Printing number " << counter << ": " << c.real << " + " << c.imaginary << "i";
    counter++;
    return out;
}
```

5.

```
const Complex Complex::PI = Complex(3.14, 0);
```

6.

Yes.

```
Printing number 0: 3.14 + 0i
Printing number 1: 2 + 1i
Printing number 2: 2 + 1i
```

Problem 3 (30 points)

1.

```
int List::size() {
    ListNode *cur = head; 1
    int count = 0;
    while(cur != NULL) { 2
        count++; 2
        cur = cur -> next; 2
    }
    return count;
}
```

2.

```
List::~List() {
    while(head != NULL) { 2
        ListNode *cur = head; 2
        head = head->next; 2
        delete cur; 2
    }
}
```

3.

This insertion sort algorithm (5 points)

insert funcion(5 points)

```
void List::insert(int value) {
    ListNode *cur = head, *prev = NULL;
    for(cur=head; cur != NULL && cur->val > value; cur = cur->next) {
        prev = cur;
    }
    ListNode *temp = new ListNode(value);
    if(prev == NULL) head = temp;
    else prev->next = temp;
    temp->next = cur;
}
```

sort function(5 points)

```
void List::insertSort() {
    List *list = new List();
    for(ListNode *cur = head; cur != NULL; cur = cur->next) {
        list->insert(cur->val);
    }
    head = list->begin();
}
```