1. (5 points) What is the difference between a class and an object?

2. (5 points) The following is a legal and working program. What does it print? You must explain your answer for any credit. Be careful, this is a trick question.

```cpp
void find_max(int nums[], int size, int &max) {
    max = nums[0];
    for (int i = 1; i < size; i++)
    {
        if (nums[i] > max)
        {
            max = nums[i];
        }
    }
}

void find_min(int nums[], int size, int min)
{
    min = nums[0];
    for (int i = 1; i < size; i++)
    {
        if (nums[i] < min)
        {
            min = nums[i];
        }
    }
}

int main()
{
    int values[] = {-42, 17, 34, 42, 99};
    // this initialized the array

    int the_max = 0;
    int the_min = 0;
    find_max(values, 5, the_max);
    find_min(values, 5, the_min);
    cout << "max = " << the_max << " min = " << the_min << endl;
    return 0; // normal exit
}
```
3. (5 points) What does the following code print? You must explain your answer for any credit.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int *i = new int;
    int *j = new int;
    *i = 42;
    *j = 42;
    if (i == j)
    {
        cout << "i == j" << endl;
    }
    else
    {
        cout << "i != j" << endl;
    }

double *x = (double *) i;
    cout << "*x = " << *x << endl;
}
```
4. (5 points) The following program consists of the five files: foo.h foo.cpp bar.h bar.cpp and main.cpp.

```cpp
// this is foo.h
class Foo
{
public:
    Foo(int value);
    void print();
private:
    int m_value;
};

// this is bar.h
class Bar
{
public:
    Bar(int value);
    void print();
private:
    Foo *my_foo;
};

// this is foo.cpp
#include "foo.h"
#include <iostream>
using namespace std;
Foo::Foo(int value)
{
    m_value = value;
}
void Foo::print()
{
    cout << m_value << endl;
}

// this is bar.cpp
#include "bar.h"
Bar::Bar(int value)
{
    my_foo = new Foo(value);
}
void Bar::print()
{
    my_foo->print();
}

// this is main.cpp
#include "foo.h"
#include "bar.h"
int main()
{
    Foo my_foo(42);
    Bar my_bar(43);
    Pop *my_foo;
    my_foo.print();
    my_bar.print();
}
```

When it is compiled the following error is printed:

```
In file included from bar.h:2,
  from main.cpp:3:
foo.h:3: error: redefinition of `class Foo'
foo.h:3: error: previous definition of `class Foo'
make: *** [main.o] Error 1
```

What is the problem? Describe the best way to fix it (don't worry about the exact syntax, just describe what you do to prevent this from happening).
5. (10 points) Given class Foo and the following code, indicate which lines of code in main() cause Foo() and/or ~Foo() to execute. For example, if you think that the line “Foo g;” causes Foo() to be executed, put Foo() on the blank line to the left of “Foo g.”

class Foo {
    public:
        Foo() {cout << "Foo::Foo() called" << endl;}
        ~Foo() {cout << "Foo::~Foo() called" << endl;}
    void print() {cout << "Foo::print_foo() called" << endl;}
};

void print_foo(Foo *my_foo)
{
    my_foo->print();
}

int main()
{
    Foo g; // declaring a new Foo object
    Foo *f;
    f = new Foo(); // dynamically instantiating a Foo object
    print_foo(new Foo()); // dynamically instantiating a Foo object
    delete f; // deleting a dynamically instantiated Foo
}

For the following 3 questions, use the code on the last page.

6. (15 points) Write the function List::find_element() that finds the element with the given index in the list. For example, if the index parameter is 1, the function should find the 1st element in the list. If the index is 2, it should find the 2nd element in the list. If the index is n, it should find the nth element. Return false if the list does not have that many elements.

bool List::find_element(int index, int &value)
{

}
7. (15 points) Write the function bool List::insert_sorted(int value) that inserts the given value into the list in such a way that the list is ordered from smallest to largest. If the number is already in the list, do not insert it and return false. If the number is not in the list, insert it and return true.

```cpp
bool List::insert_sorted(int value)
{
}
```

8. (20 points) Write a function that removes all duplicate entries in the list. Assume that the list is ordered from smallest to largest, thus all duplicates will be next to each other. For example, if the list contained \{1,2,2,2,3,3,4,5,5,5,5,6\} before calling remove_duplicates(), it should contain \{1,2,3,4,5,6\} after. Do nothing if the list is empty or contains no duplicates.

```cpp
void List::remove_duplicates()
{
}
```