1. Explain what putting `const` after a class member function means. (5 points)
   The function is not allowed to modify any class member variables.

2. If `public`, `private`, or `protected` are not used in the class member listing, which one of these are the class members? (5 points)
   `private`

3. For a single class (neither derived from nor the base class for another), explain the difference, if any, between `public` and `protected` class members with respect to how and where those members can or cannot be accessed. (5 points)
   
   `public` members can be accessed both within and outside of the class, while `protected` members can only be accessed from within the class (e.g. with a wrapper function)

4. Assume class `derived` inherits from class `base` using `protected` inheritance. If class `base` has `public`, `protected`, and `private` class members, say whether each of the different types of class members is `public`, `private`, `protected`, or `hidden` within class `derived`? (5 points)
   
   Public becomes `protected`
   Protected becomes `protected`
   Private becomes `hidden`

5. When are constructors invoked and what is their primary use? (5 points)

   Constructors are invoked when the object is first created and are typically used to initialize class member variables.

6. Explain what is wrong with the following function, assuming `classname` contains a member variable called `x`: (5 points)

   ```cpp
   void classname::fcn(int x)
   {
       this.x = x;
       return;
   }
   
   this.x should be this->x
   ```
7. List the principles of object oriented design and explain them. (15 points)

   Encapsulation - the combination of data and operations on that data (e.g. a class)
   Inheritance - the creation of new classes using previously existing classes
   Polymorphism - Using the same symbol to mean different things

8. What part of the compilation process analyzes include guards? (5 points)
   The preprocessor

9. What is wrong with the following code snippet: (5 points)
   class classname
   {
      public:
         virtual void fcn() = 0;

      private:
         int varia;
   };
   // snipped out some code... maybe
   int main()
   {
      classname cn;

      cn.fcn();
   }

   classname is an abstract class and cannot be instantiated.
10. Explain the difference between static and dynamic binding, what the default binding is for C++ programs, and how to enforce the non-default binding. (15 points)

Static binding is the default for C++ and indicates that the appropriate function calls are determined by the compiler. Dynamic binding can be enforced by the use of virtual functions and causes the appropriate function calls to be determined at run time.

The above answer is sufficient, but a better answer would also include: As an example, assume that a base class and derived class have separate implementations of a function, \texttt{fcn}, that have identical prototypes. Additionally assume a function, \texttt{call}, exists that accepts a base class reference and calls \texttt{fcn}. Finally assume that \texttt{call} is run and is passed a derived class reference. With static binding, the base class \texttt{fcn} runs, but with dynamic binding the derived class \texttt{fcn} runs.

11. Describe what a dangling pointer is and write a code snippet that creates a dangling pointer. (15 points)

A dangling pointer occurs when a pointer is pointing to a memory address that is not allocated for use by the program.

```cpp
int *a, *b;

a = new int;
b = a;
delete a; // now b is dangling and, technically, a is too!
```
12. Write the copy constructor (function only, no prototype needed, do not worry about deleting) for the following class: (25 points)

class classname
{
    // Other functions left out
    public:
        classname();
    private:
        int *p, size;
};
classname::classname()
{
    size = 10;
    p = new int[size];
    for (int i = 0; i < size; i++)
        p[i] = i;
}
classname::classname(const classname & x)
{
    size = x.size;
    p = new int[size];
    for (int i = 0; i < size; i++)
        p[i] = x.p[i];
}

13. For the code below, write one statement in each loop that will result in the two arrays having the values 0, 1, 2, ..., 8, 9 in them. For the first, use standard array notation (i.e. [ and ]). For the second, use pointer notation. (10 points)

int *p = new int[10], *q = new int[10];

for (int i = 0; i < 10; i++)
    p[i] = i;  // fill p here with array notation

for (int i = 0; i < 10; i++)
    *(q+i) = i;  // fill q here with pointer notation
14. Explain the difference between a deep copy and a shallow copy. (10 points)
In the context of pointers, a shallow copy occurs when pointers are simply copied (e.g. the memory addresses for two variables are set equal). A deep copy allocates new memory and then follows the pointers to copy all values in the location that the pointers reference.

15. Write a class (definition + implementation) called intdouble that: (40 points)
Has private member variables int i and double d
Has public member functions called set_int and set_double that each take an argument and set i and d as appropriate
Overloads the operator for addition using class member functions such that the following code is valid:
intdouble x, y, z;       z = x + y;

class intdouble
{
public:
    void set_int(int i);
    void set_double(double d);
    intdouble operator+(const intdouble & obj) const;
private:
    int i;
    double d;
};
void intdouble::set_int(int i)
{
    this->i = i;
}
void intdouble::set_double(double d)
{
    this->d = d;
}
intdouble intdouble::operator+(const intdouble & obj) const
{
    intdouble temp;
    temp.i = i + obj.i;
    temp.d = d + obj.d;
    return (temp);
}
16. Write a class definition for a class called MyClass that has two private integer member variables, width and height, and two public functions set_width and set_height that will each take one integer. Your class definition should be valid such that the following code would set the width and height as expected below. You do not need to write the implementation. (25 points)

```cpp
myclass x; x.set_width(5).set_height(10);
```

class MyClass
{
    public:
        MyClass& set_width(int i);
        MyClass& set_height(int i);
    private:
        int width, height;
};

17. What special access does a friend function have? When using information hiding, where should the friend function be written? (10 points)

Friend functions of a class are allowed to access all member variables (even private) of an object of that class type that are passed to the friend function. The implementation of the friend function should be in the same .cpp file as the regular class member functions.

18. Explain the difference between inheritance and composition. (10 points)

Inheritance involves creating a derived class from a base class, thereby inheriting the members from that base class while ostensibly adding its own new members.

Composition simply means to include an object as a member of a class (e.g. having a string object as a class member variable).
19. On the following pages, write an entire syntactically correct class that does the following: (85 points)

a. You must write valid header and implementation files.
b. Place a line between different files and write the name of the file at the top.
c. You must use include guards.
d. You do not need to write any comments.
e. The program will utilize a class to implement a two dimensional dynamically allocated array that will store multiples of a given real number.
f. Class `mults` should have three member variables:
   i. `a` (a double pointer)
   ii. `x` (an integer, the number of rows in the array)
   iii. `y` (an integer, the number of columns in the array)
   iv. You are not allowed to initialize any of these in the class definition.
g. `mults` should implement the following functions:
   i. `set_data(int x, int y, double base)`: creates the dynamically allocated two-dimensional array, `a`, of size `x`-by-`y` and fills it with multiples of `base` (as shown below with the example output)
   ii. default constructor: creates the dynamically allocated two-dimensional array, `a`, of size 3-by-3 and fills it with multiples of 1.
   iii. `display()`: display `a` in a similar way to how assignment 1 and 2 arrays were shown (multiple rows, multiple columns, appropriate spacing)
   iv. destructor: No hints here. The destructor should do what is necessary.

h. Given the following main body:

```c
int main()
{
    mults m1, m2;

    m2.set_data(2, 4, 3.4);

    m1.display();
    m2.display();

    return (0);
}
```
i. The output for the above main body should be similar to:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>6.8</td>
<td>10.2</td>
<td>13.6</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>20.4</td>
<td>23.8</td>
<td>27.2</td>
<td></td>
</tr>
</tbody>
</table>

```cpp
#include <iostream>

# ifndef _MULTS_H
# define _MULTS_H

class mults {
    public:
        mults();
        ~mults();
        void display() const;
        void set_data(int x, int y, double base);
    private:
        int x, y;
        double **a;
};
#endif
```
#include <iostream>
#include <iomanip>
#include "mults.h"

using namespace std;

mults::mults()
{
    set_data(3, 3, 1);
}

mults::~mults()
{
    for (int i = 0; i < x; i++)
        delete[] a[i];
    delete[] a;
}

void mults::display() const
{
    for (int i = 0; i < x; i++)
    {
        for (int j = 0; j < y; j++)
            cout << setw(10) << a[i][j];
        cout << endl;
    }
}

void mults::set_data(int x, int y, double base)
{
    this->x = x, this->y = y;
    a = new double*[x];
    for (int i = 0; i < x; i++)
        a[i] = new double[y];

    int count = 1;
    for (int i = 0; i < x; i++)
        for (int j = 0; j < y; j++)
            a[i][j] = base * count++;
}
1. Given that fcn2 has a void return type, takes an integer argument, and is also a member function of classname, what else can be said about the function fcn2? (5 points)

   ```cpp
   void classname::fcn(int n) const
   {
       fcn2(n);
       return;
   }
   ```

   fcn2 must also be const

2. Suppose a class will never be inherited from nor does it inherit from anything. If all member functions and variables had to be made public, private, or protected, which one makes the most sense and why? (5 points)

   public, otherwise nothing could ever be used in the class outside of automatic calls to constructors and such

3. For a single class (neither derived from nor the base class for another), explain the difference, if any, between public and private class members with respect to how and where those members can or cannot be accessed. (5 points)

   public members can be accessed both within and outside of the class, while private members can only be accessed from within the class (e.g. with a wrapper function)

4. Assume class derived inherits from class base using private inheritance. If class base has public, protected, and private class members, say whether each of the different types of class members is public, private, protected, or hidden within class derived? (5 points)

   - Public becomes private
   - Protected becomes private
   - Private becomes hidden
5. When are destructors invoked and what is their primary use? (5 points)
   Destructors are invoked when the object goes out of scope and are primarily used to delete dynamically allocated memory.

6. Explain what this is (i.e. where can it be used, syntax for its use, an example of when it is needed). (10 points)
   The `this` keyword refers to the current object and can only be used inside of a class member function. A class member variable could be accessed by using `this->member` and it is needed when returning the current object from the function or to access an object variable when a local variable also has the same name.

7. List the principles of object-oriented design and explain them. (15 points)
   - Encapsulation - the combination of data and operations on that data (e.g. a class)
   - Inheritance - the creation of new classes using previously existing classes
   - Polymorphism - Using the same symbol to mean different things

8. Write out the relevant syntax for include guards for the file `classname.h` that includes the class `classname`. In a C++ comment write where the class definition should go. (10 points)
   ```cpp
   #ifndef _CLASSNAME_H
   #define _CLASSNAME_H
   // class definition goes here
   #endif
   ```
9. Describe the difference between a virtual function and a pure virtual function, both in terms of syntax as well as how they are used differently. (10 points)

Virtual functions are used to enforce dynamic binding and are created by adding the `virtual` keyword before the function prototype in the class definition. Pure virtual functions are used to create an abstract class and utilize both the `virtual` keyword as before and = 0 at the end of the prototype.

10. Explain the difference between static and dynamic binding, what the default binding is for C++ programs, and how to enforce the non-default binding. (15 points)

Static binding is the default for C++ and indicates that the appropriate function calls are determined by the compiler. Dynamic binding can be enforced by the use of virtual functions and causes the appropriate function calls to be determined at run time.

The above answer is sufficient, but a better answer would also include:

As an example, assume that a base class and derived class have separate implementations of a function, `fcn`, that have identical prototypes. Additionally assume a function, `call`, exists that accepts a base class reference and calls `fcn`. Finally assume that `call` is run and is passed a derived class reference. With static binding, the base class `fcn` runs, but with dynamic binding the derived class `fcn` runs.

11. Describe what a dangling pointer is and write a code snippet that creates a dangling pointer. (15 points)

A dangling pointer occurs when a pointer is pointing to a memory address that is not allocated for use by the program.

```cpp
int *a, *b;

a = new int;
b = a;
delete a; // now b is dangling and, technically, a is too!
```
12. The code below causes a segfault, but only when the function is called twice, how odd!
Assuming no inheritance is used, explain the most likely cause and two ways it could be fixed --
the first by modifying the function, and the second by modifying the class itself. You do not need
to write any specific code, just explain the cause and two different solutions: (10 points)

```cpp
void fcn(classname obj, int n)
{
    obj.set_data(n);
}
```

obj likely contains dynamically allocated memory, a destructor that deletes the memory,
and does not contain a copy constructor that performs a deep copy. Therefore, when the
function finishes executing the first time, pointers are left dangling, leading to a segfault
when set_data presumably tries to set some data in the object. This can be fixed either by
rewriting the copy constructor to perform a deep copy or passing the object by reference
instead.

13. We’d like to print out all the data in this array but something weird is happening. Write the output
of the existing code, then explain what change needs to be made to the code to make it work
properly. (10 points)

```cpp
int *p = new int[4];

for (int i = 0; i < 4; i++)
    p[i] = i * 100;

for (int i = 0; i < 4; i++)
    cout << *p+i << endl;
```

Output is 0, 1, 2, 3
To correct it, change the final line to `cout << *(p + i) << endl;`
14. Assume the variable \( n \) is an \texttt{int} and has already been set. Write code that dynamically allocates an integer array, \( a1 \), of size \( n \). Next write code that shallow copies \( a1 \) to \( a2 \), and deep copies \( a1 \) to \( a3 \). (25 points)

\[
\text{int } \ast a1 = \text{new int}[n];
\]

\textbf{Shallow copy:}

\[
\text{int } \ast a2 = a1;
\]

\textbf{Deep copy:}

\[
\text{int } \ast a3 = \text{new int}[n];
\]

\[
\text{for (int } i = 0; i < n; i++)
\]

\[
\text{a3}[i] = a1[i];
\]

15. Write a class (definition + implementation) called \texttt{intdouble} that: (40 points)

- Has \texttt{private} member variables \texttt{int} \( i \) and \texttt{double} \( d \)
- Has \texttt{public} member functions called \texttt{set_int} and \texttt{set_double} that each take an argument and set \( i \) and \( d \) as appropriate
- Overloads the operator for addition using a friend function such that the following code is valid:

\[
\begin{align*}
\text{intdouble } x, y, z; & \quad z = x + y;
\end{align*}
\]

\texttt{class intdouble}

\{

\quad \text{friend intdouble } \texttt{operator+}(\text{const intdouble} & o1, \text{const intdouble} & o2);

\quad \text{public:}

\quad \quad \text{void set_int(int} i); \quad \text{void set_double(double} d); \quad

\quad \text{private:}

\quad \quad \text{int} i;

\quad \quad \text{double} d;

\};

\texttt{void intdouble::set_int(int} i)

\{

\quad \text{this}->i = i;

\}

\texttt{void intdouble::set_double(double} d)

\{

\quad \text{this}->d = d;

\}

\texttt{intdouble } \texttt{operator+}(\text{const intdouble} & o1, \text{const intdouble} & o2)

\{

\quad \text{intdouble temp;}

\quad \text{temp}.i = o1.i + o2.i;

\quad \text{temp}.d = o1.d + o2.d;

\quad \text{return} \ (\text{temp});

\}
16. Write a class definition for a class called `myclass` that has two `private` integer member variables, `width` and `height`, and two public functions `set_width` and `set_height` that will each take one integer. Your class definition should be valid such that the following code would set the width and height as expected below. You do not need to write the implementation. (25 points)

```cpp
myclass x; x.set_width(5).set_height(10);
```

class myclass
{
    public:
        myclass& set_width(int i);
        myclass& set_height(int i);
    private:
        int width, height;
};

17. Suppose `p` is a pointer that has had memory allocated to it and then it is deleted. Explain exactly what happens and what does not happen when `p` is deleted. (5 points)

The only thing that happens is that the operating system reclaims the memory, making it unavailable for use by our program. The memory address inside `p` does not change, nor is the data at the memory address pointed to by `p` changed immediately.

18. What is the purpose of information hiding and how is it accomplished? (10 points)

Information hiding is useful so that you can share your classes without allowing modification, thereby retaining interoperability. It is accomplished by writing a header file and an implementation file, then compiling the implementation file into object code. When sharing the class, give the object code and the header file away, but keep the implementation source code private.
19. On the following pages, write an entire syntactically correct class as follows: (75 points)
   a. You must write valid header and implementation files.
   b. Place a line between different files and write the name of the file at the top.
   c. You must use include guards.
   d. You do not need to write any comments.
   e. You must implement two classes, course, and student.
   f. student should have two member variables, a string for the student's name and an int for the student's grade.
   g. The default student name should be "John" and grade should be 100. You are not allowed to initialize these in the class definition.
   h. student should have wrapper functions for getting/setting each member variable and disallow names shorter than 2 characters and grades lower than 0 or higher than 100. If the name or grade is not acceptable, they should just be set to the default values.
   i. course should use composition to contain 4 students (in an array)
   j. course should implement a function called set_all that will take a string and int as parameters and set all student names & grades to those parameters.
   k. course should implement a function, display, that will display all student names and grades (one name and grade per line)
   l. Given the following main body:
      ```
      int main()
      {
       course c;
       c.display();
       c.set_all("Bob", -50);
       c.display();
       return (0);
      }
      ```
   m. The output for the above main body should be:
      ```
      John 100
      John 100
      John 100
      John 100
      Bob 100
      Bob 100
      Bob 100
      Bob 100
      ```
// course.h
#ifndef _COURSE_H
#define _COURSE_H
#include <string>
#include "student.h"
class course
{
    public:
        void set_all(string s, int i);
        void display() const;
    private:
        student a[4];
};
#endif

// course.cpp
#include "course.h"
#include "student.h"
#include <string>
#include <iostream>
void course::set_all(string s, int i)
{
    for (int i = 0; i < 4; i++)
    {
        a[i].set_name(s);
        a[i].set_grade(i);
    }
}
void course::display() const
{
    for (int i = 0; i < 4; i++)
        cout << a[i].get_name() << " " << a[i].get_grade() << endl;
}

// student.h
#ifndef _STUDENT_H
#define _STUDENT_H
#include <string>
class student
{
    public:
        student();
        void set_name(string s);
        void set_grade(int i);
        string get_name() const;
        int get_grade() const;
    private:
        string name;
int grade;
);
#endif

// student.cpp
#include "student.h"

student::student()
{
    set_name("John");
    set_grade(100);
}

void student::set_name(string s)
{
    if (s.length() < 2)
        name = "John";
    else
        name = s;
}

void student::set_grade(int i)
{
    if (i < 0 || i > 100)
        grade = 100;
    else
        grade = i;
}

string student::get_name() const
{
    return name;
}

int student::get_grade() const
{
    return grade;
}
CS 202 Midterm - 300/270 points - Fall 2017 - Dr. Williams  
Name:______________________________

Fit answers in space provided, do not write on back of pages or cram answers into margins.

1. Assuming no global variables exist, explain what is wrong with the following class member 
   function implementation: (5 points)

   ```
   void classname::fcn(int n) const
   {
       x = n;
       return;
   }
   ```

   *x must be a class member variable and const functions are not allowed to modify class 
   member variables.*

2. Why is it more common that class member functions are public while class member variables 
   are private? (5 points)

   *To prevent direct access to class member variables. We typically desire some logic (e.g. 
   error-proofing) to prevent erroneous data from being stored.*

3. For a single class (neither derived from nor the base class for another), explain the difference, if 
   any, between protected and private class members with respect to how and where those 
   members can or cannot be accessed. (5 points)

   *There is no difference.*

4. Assume class derived inherits from class base using public inheritance. If class base has 
   public, protected, and private class members, say whether each of the different types of 
   class members is public, private, protected, or hidden within class derived? (5 points)

   *Public becomes public
   Protected becomes protected
   Private becomes hidden*
5. When are destructors invoked and what is their primary use? (5 points)

Destructors are invoked when the object goes out of scope and are primarily used to delete dynamically allocated memory.

6. Suppose classname contains a private member variable called x, set the member variable to the value passed to the function: (10 points)

```cpp
void classname::fcn(int x)
{
    this->x = x; // write this line
    return;
}
```

7. List the principles of object oriented design and explain them. (15 points)

   - **Encapsulation** - the combination of data and operations on that data (e.g. a class)
   - **Inheritance** - the creation of new classes using previously existing classes
   - **Polymorphism** - Using the same symbol to mean different things

8. What are the purpose of include guards? (10 points)

   To prevent redefinitions (which would cause a compiler error) by effectively allowing headers to only be included one time.
9. Write a syntactically correct class definition that creates an abstract class called \texttt{ab} with a \texttt{void} public member function called \texttt{fcn} that takes no parameters and a \texttt{private} member integer variable called \texttt{x}. (10 points)

```cpp
class ab
{
    public:
        virtual void fcn() = 0;
    private:
        int x;
};
```

10. Explain the difference between static and dynamic binding, what the default binding is for C++ programs, and how to enforce the non-default binding. (15 points)

Static binding is the default for C++ and indicates that the appropriate function calls are determined by the compiler. Dynamic binding can be enforced by the use of virtual functions and causes the appropriate function calls to be determined at run time.

The above answer is sufficient, but a better answer would also include:

As an example, assume that a base class and derived class have separate implementations of a function, \texttt{fcn}, that have identical prototypes. Additionally assume a function, \texttt{call}, exists that accepts a base class reference and calls \texttt{fcn}. Finally assume that \texttt{call} is run and is passed a derived class reference. With static binding, the base class \texttt{fcn} runs, but with dynamic binding the derived class \texttt{fcn} runs.

11. Describe what a dangling pointer is and write a code snippet that creates a dangling pointer. (15 points)

A dangling pointer occurs when a pointer is pointing to a memory address that is not allocated for use by the program.

```cpp
int *a, *b;

a = new int;
b = a;
delete a;  // now \texttt{b} is dangling and, technically, \texttt{a} is too!
```
12. Give two instances in which a copy constructor is invoked and explain why copy constructors are written (i.e. what problem do they solve?). (10 points)

Copy constructors are invoked when creating a new object with a parameter of an existing object (e.g. `classname o2(o1)`), when returning an object from a function, or when passing an object by value to a function. Copy constructors are rewritten primarily to conduct a deep copy when objects have dynamically allocated memory.

13. Given that pointer `p` is allocated as an `int` array, write how to access the third element of `p` with both pointer and array notation. (15 points)

   `p[2]` or `*(p+2)`

14. Assume the variable `n` is an `int` and has already been set. Write code that dynamically allocates an integer array, `a1`, of size `n`. Next write code that shallow copies `a1` to `a2`, and deep copies `a1` to `a3`. (25 points)

   ```
   int *a1 = new int[n];
   Shallow copy:
   int *a2 = a1;
   Deep copy:
   int *a3 = new int[n];
   for (int i = 0; i < n; i++)
       a3[i] = a1[i];
   ```
15. Write a class (definition + implementation) called intdouble that: (40 points)

- Has private member variables int i and double d
- Has public member functions called set_int and set_double that each take an argument and set i and d as appropriate
- Overloads the operator for addition using a friend function such that the following code is valid:

```
int double x, y, z;  
z = x + y;
```

```cpp
class intdouble
{
    friend intdouble operator+(const intdouble& o1, const intdouble& o2);
public:
    void set_int(int i);
    void set_double(double d);
private:
    int i;
    double d;
};

void intdouble::set_int(int i)
{
    this->i = i;
}

void intdouble::set_double(double d)
{
    this->d = d;
}

intdouble operator+(const intdouble& o1, const intdouble& o2)
{
    intdouble temp;
    temp.i = o1.i + o2.i;
    temp.d = o1.d + o2.d;
    return (temp);
}
```
16. Write a class definition for a class called `myclass` that has two private integer member variables, `width` and `height`, and two public functions `set_width` and `set_height` that will each take one integer. Your class definition should be valid such that the following code would set the width and height as expected below. You do not need to write the implementation. (25 points)

```cpp
myclass x; x.set_width(5).set_height(10);
```

```cpp
class myclass
{
    public:
        myclass& set_width(int i);
        myclass& set_height(int i);
    private:
        int width, height;
};
```

17. What are the rules for function overloading? (5 points)

Function overloading occurs when two functions have the same name. The only rule is that each function must take different parameter lists. For example, two functions with identical names cannot both take a single integer as a parameter.

18. What are the rules for default function parameters? (5 points)

The only rule for default function parameters is that only the rightmost parameters can have default values. Once a parameter has been given a default value, the remaining parameters (on the right) must also have default values provided as well.
19. On the following pages, write an entire syntactically correct class as follows: (75 points)
   a. You must write valid header and implementation files.
   b. Place a line between different files and write the name of the file at the top.
   c. You must use include guards.
   d. You do not need to write any comments.
   e. You must implement two classes, food, and vegetable
   f. food should have a private member variable, an int for the amount of calories.
   g. food should have wrapper functions for setting/getting the calories and no other functions
   h. vegetable should inherit from food (use public inheritance)
   i. vegetable should have a private member variable called name, a string.
   j. vegetable should have a wrapper function set_data(string name, int calories) that sets the name and calories as appropriate, but if there is an attempt to set the calories to a negative value, 0 should be used instead.
   k. vegetable should have a function, display(), that prints out the name and caloric value of the food on one line.
   l. Given the following main body:
   
   ```
   int main()
   {
       vegetable v[2];

       v[0].set_data("Habanero", 20);
       v[1].set_data("Celery", -10);

       v[0].display();
       v[1].display();
       return (0);
   }
   ```
   m. The output for the above main body should be:
   
   Habanero 20
   Celery 0
/ food.h
#ifndef __FOOD_H
#define __FOOD_H

class food
{
  public:
    void set_data(int i);
    int get_data() const;
  private:
    int calories;
};
#endif

// food.cpp
#include "food.h"

void food::set_data(int i)
{
  calories = i;
}

int food::get_data() const
{
  return calories;
}
// vegetable.h
#ifndef _VEGETABLE_H
#define _VEGETABLE_H
#include <string>
#include "food.h"

class vegetable: public food
{
    public:
        void set_data(string name, int calories);
        void display() const;
    private:
        string name;
};
#endif

// vegetable.cpp
#include <string>
#include "vegetable.h"

void vegetable::set_data(string name, int calories)
{
    this->name = name;
    food::set_data(calories > 0 ? calories : 0);
}

void vegetable::display() const
{
    std::cout << name << " " << get_calories();
}