No books, no notes, and no calculators are allowed.

Question 1: ____ out of ____ points
Question 2: ____ out of ____ points
Question 3: ____ out of ____ points
Question 4: ____ out of ____ points

Total Score:

Grade:
## Question 1: Some Warm-Up Problems

Tell whether each of the following statements is true or false by checking the appropriate box. Do not check any box if you do not know the right answer, because you will lose points for incorrect answers.

<table>
<thead>
<tr>
<th>Statement</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Every C++ program is object-oriented.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>b) Every correct C program would also compile as a C++ program without any changes being necessary.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>c) Two of the main problems that we face in software engineering are complexity and change.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>d) C++ allows operator overloading.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>e) In C++, every function is a member of a class.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>f) Static class members do not belong to any particular object of their class.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>g) Const member functions cannot modify the values of their explicit inputs.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>h) Protected members are only visible to member functions of the same class.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>i) C++ programs require a main() function, which cannot be a member of any class.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>j) In order to instantiate an array of objects, the object class has to have a default constructor.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

1 point for every correct answer, -1 point for every incorrect one.
Question 2: The Crazy Programmer

The program on the following page was written by a crazy programmer. He or she produced some code that is not very elegant or robust but functional. Unfortunately, the choice of names for variables, functions, and classes is completely bizarre.

(a) For each of the following items, suggest a better name for it and describe in one or two sentences what function the item has in the program:

TuringsDog: better name: ListElement

Holds one element (a string) of a singly-linked list and a pointer to the next element.

porsche: better name: nextElement

Pointer to the next element in the list.

Bazinga: better name: OrderedList

Class that can be used to build and output an ordered list of strings.

encom: better name: firstElement

Pointer to the first element in the list.

kawoosh: better name: insertElement

Inserts a new element (string s) into the list so that the list stays in ascending alphabetical order.

evilProf: better name: deleteFirstElement

Deletes the first element in the list. If afterwards the list is empty, it returns ‘true,’ otherwise ‘false.’

glutenFree: better name: printList

Outputs the current list of strings in its alphabetical order to stdout.

1 point for every correct name + description
#include <cassert>
#include <iostream>
#include <string>

using namespace std;

struct TuringsDog
{
    string s;
    TuringsDog *porsche;
};

class Bazinga
{
public:
    Bazinga() { encom = new TuringsDog; encom->s = "zzz", encom->porsche = NULL; }
    ~Bazinga() { while (encom != NULL) evilProf(); }
    void kawoosh(string s);
    bool evilProf();
    void glutenFree();
private:
    TuringsDog *encom;
};

void Bazinga::kawoosh(string s)
{
    TuringsDog *p = encom;
    while (p->s < s)
    {
        p = p->porsche;
        TuringsDog *temp = new TuringsDog;
        temp->s = p->s;
        temp->porsche = p->porsche;
        p->s = s;
        p->porsche = temp;
    }
}

bool Bazinga::evilProf()
{
    TuringsDog *temp = encom->porsche;
    delete encom;
    encom = temp;
    return (encom == NULL);
}

void Bazinga::glutenFree()
{
    TuringsDog *temp = encom;
    while (temp != NULL)
    {
        cout << temp->s << endl;
        temp = temp->porsche;
    }
}

int main()
{
    string list[] = {"sometimes", "exam", "questions", "can", "be", "quite", "weird"};
    Bazinga b;
    for (int i = 0; i < 7; i++)
    {
        b.kawoosh(list[i]);
    }
    b.glutenFree();
    return 0;
}
(b) What is the output of the program?

be
can
exam
questions
quite
sometimes
weird
zzz

(The idea of using ‘zzz’ as a dummy last element is very inelegant and imperfect, and also printing it is clearly a bug that the crazy programmer produced.)

2 points for correct output, and 1 bonus point if you additionally got the ‘zzz’ right.

Question 3: Prime Time

On the following page you will find a C++ program that is supposed to output the first 100 prime numbers. It defines a class PrimeNumber that is initialized with an integer n. The data member “number” will be set to the smallest prime number greater or equal n. The “++” operator is overloaded to set “number” to the next greater prime number following the current one.

Unfortunately, the program contains a lot of bugs. Sometimes there are some dots ( . . . ), indicating that you have to fill in some code there. But you may have to fill in code in other places, too.

Good luck!
```cpp
#include <iostream>
using namespace std;

class PrimeNumber
{
public:
    int PrimeNumber(int n): number(n - 1) { (*this)++; }  // delete
    bool IsMultipleOf(int n) { return (number%n == 0); }  
    PrimeNumber operator++()  
        { do number++; while (!IsPrime()); return *this; }  // delete
    operator int() { return number; }  

private:
    bool IsPrime();  
    int number;
};

void PrimeNumber::IsPrime()
{
    for (int i = 1; i <= number/2; i++)  
        if (IsMultipleOf(i))  
            return false;  

    return true;
}

int main()
{
    int n;
    PrimeNumber* p(2);  
    for (int i = 1; i <= 100; i++)  
    {  
        n = p;  
        cout << n << " ";  
        p++;  
    }
    cout << endl;
    return 0;
}

1 point for every bug you found. No penalty for marking bugs that actually are no bugs.
```
Question 4: Testing
What is test-driven development? Why would a software company want to use it?

Some of the main points to mention here:

- Write tests before writing the actual functions.
- Leads to short development/refactoring cycles and frequently tested code.
- These short cycles increase flexibility and productivity.

Maximum of 5 points.