Instructions:

- Print your name in the space provided below.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that you have done so, and label the continuation with the question number.
- If you want partial credit, justify your answers briefly and concisely, even when justification is not explicitly required.
- There are 14 questions, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- This is a closed-book, closed-notes examination. No calculators or other electronic devices may be used during this examination. You may not discuss (in any form: written, verbal or electronic) the content of this examination with any student who has not taken it. You must return this test form when you complete the examination. Failure to adhere to any of these restrictions is an Honor Code violation.

Do not start the test until instructed to do so!

Name ____________________________

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

__________________________________________

signed
For the next two questions, consider the following set template class, from project 2:

```cpp
template <class elem> class SetT {
    private:
        string name;
        int size;
        elem* elements; //dynamic array of set elems
    public:
        SetT();
        SetT(const elem& e);
        SetT(const SetT<elem>& src);
        bool operator==(const SetT<elem>& RHS) const;
        bool operator!=(const SetT<elem>& RHS) const;
        bool operator<=(const SetT<elem>& RHS) const;
        bool operator<(const SetT<elem>& RHS) const;
        bool operator>=(const SetT<elem>& RHS) const;
        bool operator>(const SetT<elem>& RHS) const;
        SetT<elem> operator||(const SetT<elem>& RHS) const;
        SetT<elem> operator&&(const SetT<elem>& RHS) const;
        int operator+() const;
        SetT<elem>& operator=(const SetT<elem>& RHS);
        SetT<elem>& operator=(const string& sname);
        operator string();
        ~SetT();

        // friend operators:
        friend ostream& operator<<(ostream& Out, const SetT<elem>& toPrint);
        friend bool operator&&(elem LHS, const SetT<elem>& RHS);
    }

#include "Set.cpp"
```

1. [5 points] Assuming all of the member functions of the template class SetT have been implemented correctly, circle any of the following statements that would not compile.

```cpp
SetT<float> fSet1(1.0F), fSet2(2.0F), fSet3(3.0F), fSet4, fSet5;

fSet1 = fSet1 + -1.0F;

fSet3 = (fSet2 + -2.0F + -3.0F);

int s = +(fSet4 && fSet3 || fSet2);

fSet5 = +fSet3 + fSet2 + (-5.0F);
```
2. [5 points] Assume that a set of strings is instantiated:

```cpp
SetT<string> strSet("");
```
Is the relationship between `SetT` and `string` an association, aggregation or inheritance? Very briefly justify your answer.

3. [15 points] The difference of two sets, \((A - B)\), is defined as the set containing all of the elements in the first set, \(A\), that are not present in the second set, \(B\). Assume that all of the other `SetT` member functions have been implemented. Give the code to overload the C++ subtraction operator to implement the set difference operation. (Note: The underlying dynamic array of elements must be used for the set storage.)
5. [10 points] In addition to overloading the C++ subtraction operator to implement the set difference operation, for a more robust set type the subtraction operator should also be overloaded to perform element deletion, (i.e. removal of an element from a set). Give the function prototype, (and only the prototype – not the implementation), to overload the C++ subtraction operator to represent set element deletion. Give a second prototype to make the operator for element deletion commutative.

Consider the following classes:

```cpp
class Oper {
private:
   int Op;
public:
   Oper(int V = 5) {Op = V;}
   int getOp() const {return Op;}
   Oper& setOp(int R)
   {Op = R; return *this;}
};

class Sum {
private:
   Oper LHS, RHS;
   int S;
public:
   Sum() {S = LHS.getOp() + RHS.getOp();}
   Oper& Lop() {return LHS;}
   Oper& Rop() {return RHS;}
   Oper Exec() {return Oper(LHS.getOp() + RHS.getOp());}
};
```

5. [5 points] Consider the declaration: `Sum T;`

Is it possible to say what value `T.S` will have? If yes, briefly explain what that value is and why. If no, briefly explain why not.

6. [10 points] Given the following code segment to execute:

```cpp
Sum add;
Oper op1(-1), op2(-2), ope;
add.Lop() = op2;
op1 = add.Rop();
ope = add.Exec().setOp(0);
cout << op1.getOp() << endl;
cout << add.Exec().getOp() << endl;
cout << ope.getOp() << endl;
```

What is output by the above code? (If the code will not compile or will produce an execution error, indicate which and briefly explain why.)
For the next 6 questions, consider the following classes:

class P {
private:
  int A, B;
public:
  P(int a = 0, int b = 0) {A = a; B = b;}
  int getA() const {return A;}
  int getB() const {return B;}
  void setAB(int a = 0, int b = 0) { A = a; B = b;}
~P() {};
};

class Q {
private:
  P C;
  int D;
public:
  Q(P c = P(-1, -1), int d = -2) {C = c; D = d;}
  Q& setC(P c) {C = c; return *this;}
  Q& setD(int d) {D = d; return *this;}
  P getC() const {return C;}
  int getD() const {return D;}
  int Tot() const {return (D + C.getA() + C.getB());}
~Q() {};
};

class R : public Q {
private:
  int E;
public:
  R(int e) {E = e;}
  R(Q q, int e) : Q(q) {E = e;}
  R& setE(int e) {E = e;}
  int getE() const {return E;}
  int Xprod() const {
    return (E * getD() * getC().getA() * getC().getB());
  }
  int Avg() const {
    return ( (E + getD() + getC().getA() + getC().getB()) / 4 );
  }
};

Given the following declaration:

    R r(Q(P(-3,-3),-3), -3);

7. [6 points] Can one make the following member invocation, (very briefly explain)?

    int i = r.Tot();

8. [6 points] Can one make the following member invocation, (very briefly explain)?

    r.setAB(-7, -7);
For the next question, consider the following alternative implementation of \( \text{R::Avg()} \), (replacing the current implementation in the class declaration):

```cpp
int Avg() const {
    return (E + Tot()) / 4;
}
```

9. [6 points] Briefly explain why this alternative implementation is valid or invalid.

For the next 2 questions, consider the following alternative implementation of \( \text{R::Xprod()} \), (replacing the current implementation in the class declaration):

```cpp
int Xprod() const {
    return (E * D * C.getA() * C.getB());
}
```

10. [6 points] Briefly explain why this alternative implementation is invalid.

11. [6 points] Without eliminating inheritance, what change would make this alternative implementation valid, (and without changing the alternative implementation code itself)?

12. [6 points] Give the output of the following code:

```cpp
R r2(-11);
cout << r2.getD() << r2.getC().getA() << r2.getC().getB() << endl;
```
For the next two questions, consider the following function:

```cpp
bool test(bool b) throw(exception) {
    if (b) {
        throw exception(); //throw a standard exception object
        return ( !b );
    }
}
```

13. [8 points] Given that the runtime stack contains the following function activations records:

```
<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
</tr>
<tr>
<td>fn3</td>
</tr>
<tr>
<td>fn2</td>
</tr>
<tr>
<td>fn1</td>
</tr>
<tr>
<td>main</td>
</tr>
</tbody>
</table>
```

Briefly explain what happens to the program and function execution and objects if \( b == true \) when \test()\ is invoked?

14. [6 points] Assuming that \( b == true \) when \test()\ is invoked, briefly explain when the \return\ statement in \test()\ will be executed?