Programming Languages
Homework 3

1. [30] (note numbers at the start of a problem indicate how many points the problem is worth in grading.) The C++ template below defines a class for Vectors of any type T. Implement the member functions properly and follow the explicit naming conventions shown in the template below. You may copy and paste this template definition into your solution; it is highly suggested that you work from the template. For Vector assignment, reallocate the left Vector to be the same size as the right Vector. Also, be sure your copy constructor makes a *deep copy* of the vector. Write the entire program in one header file called **vector.h**.

```
#ifndef VECTOR_H
#define VECTOR_H
#include <iostream>
using namespace std;
template
     <typename T> // Assume Vector only takes in int or double for T
class Vector {
private:
     int sz;   // the number of elements in this Vector
     T* buf;   // the base of the array of Ts, you must allocate it
public:
     Vector(int n) // Vector v1(10); -- create a 10 element Vector
     {
          // Implementation Here;
     }
     Vector(initializer_list<T> L) // Vector v1{T1, T2, T3};
     {
          // Implementation Here;
     }
     ~Vector() /* destructor called automatically when a Vector dies
     {
          // Implementation Here;
     }
     Destructor should free memory used. your program should have no
     leak/lost/still-reachable/errors(suppressed or not), besides
     72704 bytes in one still-reachable block (a g++/valgrind bug on
     some versions). */
     Vector(const Vector & v) // Vector v2(v1); deep-copy
     {
```

```
     // Implementation Here;
}
int size() const // v1.size() returns 10 for v1 example above
{
     // Implementation Here;
}
T & operator [] (const int i) /* T x = V[i];
{
     // Implementation Here;
}
Access out-of-bound index should throw an error to be caught in
outside scope */
T operator * (const Vector & v) const
{
     // Implementation Here;
}
// T x = V1 * V2; dot product
// e.g. [1, 2] * [3, 4, 5] = 1 * 3 + 2 * 4 + 0 = 11
// Assume an empty Vector will cause the product to be 0.
Vector operator + (const Vector & v) const
// V3 = V1 + V2; [1, 2, 3] + [4, 5, 6, 7] = [5, 7, 9, 7]
{
     // Implementation Here;
}
const Vector & operator = (const Vector & v) // V1 = V2;
{
     // Implementation Here;
}
bool operator == (const Vector & v) const // if (V1 == V2)...
{
     // Implementation Here;
}
bool operator != (const Vector & v) const // if (V1 != V2)...
{
     // Implementation Here;
}
friend Vector operator * (const int n, const Vector & v)
// V1 = 20 * V2; -- each element of V1 is element of V2 * 20
{
     // Implementation Here;
}
```

```
    friend Vector operator + (const int n, const Vector & v)
    // V1 = 20 + V2; -- each element of V1 is element of V2 + 20
    {
        // Implementation Here;
    }
    friend ostream& operator << (ostream & o, const Vector & v)
    // cout << V2; -- prints the vector in this format
    // (v0, v1, v2, ... vn-1);
    {
        // Implementation Here;
    }
};
#endif
```

To test your Vector class works, create a separate file named **test.cpp** and write a main() function that tests each of the member functions you implemented. Below we have given an example of what you might want to test.

NOTE: This is how we will be testing your submission, so ensure that you maintain the names of the class and member functions from above.

```
#include "vector.h"
int main() // I'll start it for you
{
    Vector<int> intVec{1,3,5,7,9};
    Vector<double> doubleVec{1.5,2.5,3.5,4.5};
    Vector<int> iv(intVec);
    Vector<double> dv(doubleVec);
    cout << "intVec" << intVec << endl;
    // "intVec(1, 3, 5, 7, 9)"
    cout << "iv" << iv << endl;
    // "iv(1, 3, 5, 7, 9)"
    cout << "doubleVec" << doubleVec << endl;
    // "doubleVec(1.5, 2.5, 3.5, 4.5)"
    cout << "dv" << dv << endl;
    // "dv(1.5, 2.5, 3.5, 4.5)"

    // add at least one test case for each method defined in Vector
    return 0;
}
```

2. [30] Short answer questions about your program in Problem 1:

a. [5] What is the meaning of const after a member function prototype?
b. [5] What happens if you use the default copy constructor for Vector?
c. [5] What happens if you use the default assignment operator for Vector?
d. [5] Why pass Vector by reference but make it const as with operator *?
e. [5] Why are operators *, +, and << friends and not member functions?
f. [5] Why does operator [] return a T & as opposed to a T?

3. [40] Show the output of the following program (written in a hypothetical Ada-like language) executed twice: **1) assuming static scoping**, and **2) assuming dynamic scoping**. Assume that appropriate 'put' subroutines are defined to print out their arguments in a nice format.
NOTE: Be sure you can execute this type of problem with mental tracing and drawing pictures of memory because you will do it several more times on a quiz and on the final quiz.

```
PROCEDURE Simple_Scoping IS
     m: integer;
     PROCEDURE P IS
     BEGIN
          m := 12;
     END P;
     PROCEDURE Q IS
          m : integer;
     BEGIN
          m := 6;
          P;
          put("In Q m = ", m);
     END Q;
BEGIN
     m := 10;
     put("In Simple_Scoping Initially   m = ", m);
     Q;
     put("In Simple_Scoping after Q   m = ", m);
     P;
     put("In Simple_Scoping after P   m = ", m);
END Simple_Scoping;
```

4. Submit two files to gradescope (each file will have its own submission):

- **a report showing:**
    a. answers to the short questions (**give the question first** – I suggest you copy them from this assignment, paste into your report, then type up your answers).
    b. your output for both mental trace runs of Simple_Scoping.

  NOTE: Please **type all output** (problems 2 & 3) instead of handwriting, and remember to **map your report** to problems sections on GradeScope.

- **a single file vector.h containing your full definition of class Vector. As a note: the autograder will be using g++ v7.4.0 with the flag -std=c++11.**

TO TEST YOUR CODE:
Run the following command (assuming you have followed all naming conventions)

g++ test.cpp -o test