Lab 8

Due No Due Date **Points** 10 **Available** after Feb 24 at 7pm

Templates and STL

In order to get credit for the lab, you need to be checked off by the end of lab. You can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish before the next lab. For extenuating circumstances, contact your lab TAs and the instructor.

For this lab we will make our own templated vector class, and then compare it with the STL vector class.

(3 pts) Implementing the Big Three for the Vector class

Copy the vector.hpp and try_vector.cpp files below. Note that this current implementation doesn't have a function for capacity().

vector.hpp

```
#include <stdlib.h>
template <class T>
class vector {
   private:
      T *v;
      int s;
   public:
      vector(){
             s=0;
             v=NULL;
      }

      ~vector(){
             delete [] v;
      }

      int size() {
             return s;
      }

      void push_back(T ele) {
             T *temp;
             temp = new T[++s];
             for(int i=0; i<s-1; i++)
                temp[i]=v[i];
             delete [] v;
             v=temp;
             v[s-1]=ele;
```

```
    }
};
```

try_vector.cpp

```
#include "./vector.hpp"
#include <vector>
#include <iostream>

//We do not want to include either statement. We wouldn't be able to compare our vector template to the Stand
ard
//using namespace std;
//using std::vector;
using std::cout;
using std::endl;

int main (){
   vector<int> v;   //Our vector class
   std::vector<int> stdv; //Standard vector

   //Compare operation of our vector to std
   v.push_back(23);
   stdv.push_back(23);

   cout << "Our vector size: " << v.size() << endl;
   cout << "STL vector size: " << stdv.size() << endl;

   return 0;
}
```

You need to finish implementing the Big Three for your vector template class, since vectors use dynamic memory.

Copy Constructor

```
vector(vector<T> &other){...}
```

Assignment Operator Overload

```
void operator=(vector<T> &other){...}
```

(1 pt) Testing Your Template

Convince yourself and the TAs that your templated vector class constructors and Big Three are working as intended. Modify the try_vector.cpp file as needed to show this.

(3 pts) Adding Functions

After you've got the previous section working, it is time to move forward towards implementing the rest of your vector template class. Two key functions for vectors are used for accessing the elements. Implement the functions below:

```
T operator[](int); //Only performs address arithmetic
T at(int);         //Check to make sure not out of bounds
```

(1 pt) Testing Functions

Convince yourself and the TAs that both of your templated vector class functions for accessing data work. Modify the try_vector.cpp file as needed to show this.

(2 pt) Rounding out the Vector class

How would this change for having capacity and size members?

What will you have to change in your vector class when you add a capacity private member? The capacity is the actual number of elements allocated on the heap for the vector, and the size is the number that is being used.

How would the constructors and push_back() function change?

Write out a plan for the extra constructors you might need for testing this and how the push_back() function changes with regard to having both capacity and size members.

Implement these extra constructors and change your push_back() function to operate correctly with capacity and size, now that they may differ.

Include a resize() function.