ECS 40 **Program #7 (50 pts, 2 hours)** **Spring 2016**

Due:Wednesday, May 18th, 11:59 PM Executable name: templates.out Handin to cs40a p7

Filenames: authors.csv, Makefile, main.cpp, vector.cpp, vector.h

New concepts: strings, templates, iterators.

For this assignment, you will be creating a templated Vector class, and an associated iterator class. The char version of your class will exactly mimic the STL string methods. Your Vector class will be similar to that you wrote for Program #6. Because it is a template, when you write the code for the char version that mimics string functions, you will be writing the code for the int version at the same time! The capacity should be named storage, and size named count, so the compiler will let you write the size(), and capacity() methods without conflicting with a data member's name.

Note that charToInt() appends a zero to the array of integers it creates so that you can use 0 as a sentinel value for the end of a list in an array in your templated class. Your Vector<char> should match the output of the string exactly. You can learn more about each function at: http://www.cplusplus.com/reference/string/string/#types You may assume that all values used will be valid, albeit finds can be unfound. You will find main.cpp, my templates.out, and templateTest1.txt in ~ssdavis/40/p7

Specifications, and order of development. Use the string methods to guide the actions of your methods. Uncomment each section of main() as you write your methods in Vector. I implemented them in the order in which they are presented.

1. Write a templated Vector class with size(), capacity(), and a stub for the overloaded operator<<.
   1.1. Remember to use the magic incantation everywhere: "`template <typename T>`"
   1.2. http://web.mst.edu/~nmjxv3/articles/templates.html gives guidance for declaring a friend function in a template class. You should use approach #2. Note that "<typename T>" and "<class T>" can be used interchangeably in almost all situations. Note that you have to provide a forward declaration for the Vector class (complete with magic incantation) above the declaration of the operator in the header file.
2. Write the default constructor, and then destructor.
3. If you look at main you will see that an array of elements is often passed as a parameter to methods. While you could write methods that take arrays as their parameters, there is a way to make the methods so that they would accept either a const T array, or a const Vector<T> as their parameter, and thus be more flexible. Write a constructor that takes a const array of T as its parameter. Now write an assignment operator that takes a Vector<T> as its parameter. When you compile you will find that the compiler will be happy to implicitly call your array based constructor for such lines as "`vectorC2 = s;`".
4. Write the actual code for the overloaded<<. Your code should now handle the assignment operation properly.
5. Uncomment main(), write, compile, and test each method before moving to the next one in the following order.
   5.1. operator+=, find(), substr(), insert(), and then replace(). If there is not enough storage, then this resizes to the exact space needed.
   5.2. find(). When string::find() fails to locate the specified string it returns string::npos, which is defined as the largest possible value of size_t (the return type of find). You can create this value with "`size_t - 1;`"
   5.3. substr(). Since string::substr() returns a string, yours will have to return a Vector<T>. Note that it cannot be a reference since the substring is originally created in a Vector local to the function, and thus will be destroyed by the destructor before the assignment operator has a chance to copy it. This will require you to write a copy constructor.
   5.4. insert(). This is the trickiest of all the functions. You must double the current storage until it's largest enough to hold the result. Take time to think about how you are going to move the original values to the right so that they are not overwritten.
   5.5. replace(). This is also tricky, but writing insert() should have taught you some lessons. This follows the same storage rules as insert(). You may wish to copy insert(), and then modify that. You have to account for the fact that sometimes the count actually shrinks!
6. Forward Iterators.
   6.1. You will need to create a templated VectorIterator class that is a friend of the Vector<T> class.
      6.1.1. As with the ListNode class, you will have make a forward declaration of the class Vector<T>, but you should have done that for the overloaded operator<< for Vector anyway.
      6.1.2. The VectorIterator class has a pointer to a Vector<T> object named vectorPtr, and an int position.
      6.1.3. The class has "standard constructor" that can do double duty as a default constructor if you provide default values for both of its parameters.
      6.1.4. The class also has overloaded ++, !=, and * operators.
   6.2. The secret here is to do a public "`typedef VectorIterator<T> iterator;`" in the Vector class.
   6.3. Vector::end() returns a VectorIterator that has its position set to -1.

6.4. By looking how the string iterator works, you should be able to figure out Vector::begin(), and the overloaded operators for VectorIterator. Please try not post to Piazza asking what they do, or how they do it. Only the operator++ is more than one line long. Try to think of them as little puzzles. You will note that main() never tries to dereference an invalid iterator, e.g. end(). Just make main() work. Don't worry about handling errors that occur in use outside of main().

7. Reverse Iterators
   7.1. You will need to create a templated ReverseVectorIterator class that is only slightly different from the VectorIterator class.
   7.2. Just copy and paste everything the you wrote for VectorIterator class and Vector class, and make the appropriate changes.
8. All methods must have the appropriate const declarations. Please list private, then public in classes.
9. Code must be submitted by exactly one member of each team. Double submissions, or errors in the authors.csv format will result in the team losing five points. Your handin command line will be:
   handin cs40a p7 authors.csv Makefile vector.h vector.cpp main.cpp

```
int main(int argc, char** argv)
{
  string str, str2;
  char operation, s[ARRAY_SIZE];
  int nums[ARRAY_SIZE], start, length, subStart, subLength;
  string::iterator sitr;
  string::reverse_iterator sRitr;
//  Vector<char>::iterator vcItr;
//  Vector<int>::iterator viItr;
//  Vector<char>::reverse_iterator vcRitr;
//  Vector<int>::reverse_iterator viRitr;
  Vector<char> vectorC, vectorC2;
  Vector<int> vectorI, vectorI2;
  ifstream inf(argv[1]);

  while(inf >> operation)
  {
    switch(operation)
    {
      case 'A' :
        cout << "\nAssignment operator:\n";
        inf.getline(s, ARRAY_SIZE);
        charsToInts(s, nums);
        str = str2 = s;
        str = str;
        cout << str.size() << ' ' << str.capacity() << ' ' << str << endl;
        vectorC = vectorC2 = s;
        vectorC = vectorC;
        cout << vectorC.size() << ' ' << vectorC.capacity() << ' ' << vectorC <<
endl;
        vectorI =  vectorI2 = nums;
        vectorI = vectorI;
        cout << vectorI.size() << ' ' << vectorI.capacity() << ' ' << vectorI <<
endl;
        break;
      case '+' :
        cout << "+= operator:\n";
        inf.getline(s, ARRAY_SIZE);
        charsToInts(s, nums);
        str += s;
 //       vectorC += s;
 //       vectorI += nums;
```

```
        cout << str.size() << ' ' << str.capacity() << ' ' << str << endl;
        cout << vectorC.size() << ' ' << vectorC.capacity() << ' ' << vectorC <<
endl;
        cout << vectorI.size() << ' ' << vectorI.capacity() << ' '  << vectorI <<
endl;
        break;
      case 'F' :
        cout << "\nFind:\n";
        inf.getline(s, ARRAY_SIZE);
        charsToInts(s, nums);
        cout << str.find(s) << endl;
//        cout << vectorC.find(s) << endl;
//        cout << vectorI.find(nums) << endl;
        break;
      case 'S' :
        cout << "\nSubstring:\n";
        inf >> start >> length;
        str2 = str.substr(start, length);
//        vectorC2 = vectorC.substr(start, length);
//        vectorI2 =  vectorI.substr(start, length);
        cout << str2 << endl;
        cout << vectorC2 << endl;
        cout << vectorI2 << endl;
        break;
      case 'I' :
        cout << "\nInsert:\n";
        inf >> start;
        inf.getline(s, ARRAY_SIZE);
        charsToInts(s, nums);
        str.insert(start, s);
 //       vectorC.insert(start, s);
 //       vectorI.insert(start, nums);
        cout  << str.size() << ' ' << str.capacity() << ' '  << str <<endl;
        cout  << vectorC.size() << ' ' << vectorC.capacity() << ' ' <<  vectorC <<
endl;
        cout  << vectorI.size() << ' ' << vectorI.capacity() << ' ' << vectorI <<
endl;
        break;
      case 'R' :
        cout << "\nReplace:\n";
        inf >> start >> length >> subStart >> subLength;
        inf.getline(s, ARRAY_SIZE);
        charsToInts(s, nums);
        str.replace(start, length, s, subStart, subLength);
//        vectorC.replace(start, length, s, subStart, subLength);
//        vectorI.replace(start, length, nums, subStart, subLength);
        cout << str.size() << ' ' << str.capacity() << ' ' << str << endl;
        cout << vectorC.size() << ' ' << vectorC.capacity() << ' ' << vectorC <<
endl;
        cout  << vectorI.size() << ' ' << vectorI.capacity() << ' ' << vectorI <<
endl;
        break;
      case 'L' :
        cout << "\nForward iterator:\n";
        for(sitr = str.begin(); sitr != str.end(); sitr++)
          cout << *sitr;
```

```
        cout << endl;

//        for( vcItr = vectorC.begin(); vcItr != vectorC.end(); vcItr++)
//          cout << *vcItr;

        cout << endl;

//        for( viItr = vectorI.begin(); viItr != vectorI.end(); viItr++)
//          cout << *viItr;

        cout << endl;

        break;
      case 'M' :
        cout << "\nReverse iterator:\n";
        for(sRitr = str.rbegin(); sRitr != str.rend(); sRitr++)
          cout << *sRitr;

        cout << endl;

//        for( vcRitr = vectorC.rbegin(); vcRitr != vectorC.rend(); vcRitr++)
//          cout << *vcRitr;

        cout << endl;

//        for( viRitr = vectorI.rbegin(); viRitr != vectorI.rend(); viRitr++)
//          cout << *viRitr;

        cout << endl;
        break;
    } // switch
  }  // while more in file
  return 0;
} // main())

[ssdavis@lect1 p7]$ cat templateTest1.txt
AShort test.
AVery long test to see if the size and capacity are correctly stored and
available.
AYour reasoning is excellent,
+ it's only your assumptions that are wrong.
Fonly your
Fits
S 30 8
I 43 basic
+ Whatever became of eternal truth?
R 39 4 6 2their myopia
R 76 26 32 92Education has so much to learn! If you learn one useless thing every
day, in a single year you'll learn 365 useless things. Oh well.
L
M
[ssdavis@lect1 p7]$ templates.out templateTest1.txt

Assignment operator:
11 11 Short test.
11 11 Short test.
11 11 583604611614616532616601615616546
```

Assignment operator:
82 82 Very long test to see if the size and capacity are correctly stored and available.
82 82 Very long test to see if the size and capacity are correctly stored and available.
82 82
586601614621532608611610603532616601615616532616611532615601601532605602532616604601532615605622601532597610600532599597612597599605616621532597614601532599611614614601599616608621532615616611614601600532597610600532597618597605608597598608601546

Assignment operator:
28 28 Your reasoning is excellent,
28 28 Your reasoning is excellent,
28 28
589611617614532614601597615611610605610603532605615532601620599601608608601610616544
+= operator:
71 71 Your reasoning is excellent, it's only your assumptions that are wrong.
71 71 Your reasoning is excellent, it's only your assumptions that are wrong.
71 71
589611617614532614601597615611610605610603532605615532601620599601608608601610616544532605616539615532611610608621532621611617614532597615615617609612616605611610615532616604597616532597614601532619614611610603546

Find:
34
34
34

Find:
18446744073709551615
18446744073709551615
18446744073709551615

Substring:
t's only
t's only
616539615532611610608621

Insert:
77 142 Your reasoning is excellent, it's only your basic assumptions that are wrong.
77 142 Your reasoning is excellent, it's only your basic assumptions that are wrong.
77 142
589611617614532614601597615611610605610603532605615532601620599601608608601610616544532605616539615532611610608621532621611617614532598597615605599532597615615617609612616605611610615532616604597616532597614601532619614611610603546
+= operator:
111 142 Your reasoning is excellent, it's only your basic assumptions that are wrong. Whatever became of eternal truth?
111 142 Your reasoning is excellent, it's only your basic assumptions that are wrong. Whatever became of eternal truth?
111 142
589611617614532614601597615611610605610603532605615532601620599601608608601610616544532605616539615532611610608621532621611617614532598597615605599532597615615617

961261660561161061553261660459761653259761460153261961461161060354653258760459761660161860161453259860159959760960153261160253260161660161461059760853261661461761660456304563

Replace:
109 142 Your reasoning is excellent, it's only my basic assumptions that are wrong. Whatever became of eternal truth?
109 142 Your reasoning is excellent, it's only my basic assumptions that are wrong. Whatever became of eternal truth?
109 142
589611617614532614601597615611610605610603532605615532601620599601608608601610616544532605616539615532611610608621532609621532598597615605599532597615615617609612616605611610615532616604597616532597614601532619614611610603546532587604597616601618601614532598601599597609601532611602532601616601614610597608532616614617616604563

Replace:
175 284 Your reasoning is excellent, it's only my basic assumptions that are wrong. If you learn one useless thing every day, in a single year you'll learn 365 useless things. truth?
175 284 Your reasoning is excellent, it's only my basic assumptions that are wrong. If you learn one useless thing every day, in a single year you'll learn 365 useless things. truth?
175 284
589611617614532614601597615611610605610603532605615532601620599601608608601610616544532605616539615532611610608621532609621532598597615605599532597615615617609612616605611610615532616604597616532597614601532619614611610603546532573602532621611617532608601597614610532611610601532617615601608601615615532616604605610603615546532532616614617616604563

Forward iterator:
Your reasoning is excellent, it's only my basic assumptions that are wrong. If you learn one useless thing every day, in a single year you'll learn 365 useless things. truth?
Your reasoning is excellent, it's only my basic assumptions that are wrong. If you learn one useless thing every day, in a single year you'll learn 365 useless things. truth?
589611617614532614601597615611610605610603532605615532601620599601608608601610616544532605616539615532611610608621532609621532598597615605599532597615615617609612616605611610615532616604597616532597614601532619614611610603546532573602532621611617532608601597614610532611610601532617615601608601615615532616604605610603615546532532616614617616604563

Reverse iterator:
?hturt .sgniht sselesu 563 nrael ll'uoy raey elgnis a ni ,yad yreve gniht sselesu eno nrael uoy fI .gnorw era taht snoitpmussa cisab ym ylno s'ti ,tnellecxe si gninosaer ruoY
?hturt .sgniht sselesu 563 nrael ll'uoy raey elgnis a ni ,yad yreve gniht sselesu eno nrael uoy fI .gnorw era taht snoitpmussa cisab ym ylno s'ti ,tnellecxe si gninosaer ruoY
563604616617614616532532546615603610605604616532615615601608601615617532553554551532610614597601608532608608539617611621532614597601621532601608603610605615532597532610605532544621597600532621614601618601532603610605604616532615615601608601615617532601610611532610614597601608532617611621532602573532546603610611614619532601614

```
97532616597604616532615610611605616612609617615615597532599605615597598532621609532621608610611532615539616605532544616610601608608601599620601532615605532603610605610611615597601614532614617611589
[ssdavis@lect1 p7]$
```