CS2134 Homework Assignment 4
Fall 2016
Due* 11:00 p.m. Sat. Oct 8, 2016

October 1, 2016

Assignment 4 include a programming portion and a written portion. The programming portion must compile and consist of a single file (hw04.cpp), and the type written portion should consist of a single file (hw04written) in a .pdf format. Be sure to include your name at the beginning of each file! You must hand in both files via NYU Classes.

Programming Part:

1. Add the method `erase( Vector<Object>::iterator vItr)` to the `Vector` class. The signature of your method should be:

   ```
   iterator erase( iterator vItr)
   ```

2. Write a **recursive**[1] function that returns the sum of the digits of a positive integer. The prototype for this function is:

   ```
   int sumDigits(int x);
   ```

   Remember to include a base case.

   If $x = 519$ then the function returns 15. (i.e. $15 = 5 + 1 + 9$.)
   If $x = 312$ then the function returns 6. (i.e. $312 = 3 + 1 + 2$.)

   This function should not have any loops.

   You may extend your algorithm to allow for the input to be a negative integer.
   If $x = -4$ then the function returns 4.
   If $x = -123$ then the function returns $6 = 1 + 2 + 3$.

*A bonus of %10 percent will be given if you turn in this homework assignment by Fri. Oct 7 at 11:00 p.m.
[1]This problem would better solved non-recursively, but it for you to learn how to write a recursive function.

3. Write a program that sums the items in a vector using a *divide and conquer* algorithm. Create a driver function to call the recursive function; the driver function returns the value returned by the recursive[2] function.

   The driver function takes one parameter of type `vector<int>` and returns an `int`. The prototype for driver function is:

   ```
   int sumVector( const vector<int> & a) // driver program
   ```

   The recursive function's parameters should be iterators that signify a range `[left, right)`.

   Remember to include a base case.

   If a contains $\{-9, 12, 8\}$ then the function returns 11. (i.e. $11 = -9 + 12 + 8$.)
   If a contains $\{3\}$ then the function returns 3.
   If a contains $\{-21, 31, 14, 1, -20\}$ then the function returns 5. (i.e. $5 = -21+31+14+1-20$.)

   The function should:
   Divide the vector in half.
   Recurse on the left hand side.
   Recurse on the right hand side.
   Return the sum of both sides.

[2] This problem would better solved non-recursively, but it for you to learn how to write a recursive function.

Written Part

1. Using big-Oh notation, give the worst case run time for the method `erase`, which you implemented programming problem 1.

2. Using big-Oh notation, give the worst case run time for the divide and conquer function your wrote in programming question `dvdandcnqr`.

3. For the `Vector` class, the method `erase`, (which you implemented programming problem 1), potentially invalidates an iterator (i.e.We will say an iterator is still "valid" if it refers to the same item as before the method was called.). State which iterators are still valid and which iterators are no longer valid. Give your answer as a range of iterators.

4. For the `Vector` class we implemented in class, and for the following code snippet:

```
Vector<int> c(1);
Vector<int>::iterator itr = c.begin();
for (int i = 0; i < 100; i++)
  c.push_back(i);

c[0] = 10;

cout << *itr << endl;
```

   What is printed? I.e. Does `itr` contain the address where the value `c[0]` is stored? Explain your answer.

5. For the `Vector` class we discussed in class, if we removed the word `explicit` in front of the constructor, i.e.

```
template <class Object>
class Vector
{
public:
    Vector( int initSize = 0 ) :
        theSize( initSize ), theCapacity( initSize + SPARE_CAPACITY )
        { objects = new Object[ theCapacity ]; }

        // ... rest of class the same as before
 }
```

   What would be printed by the following code snippet?

```
Vector<int> v(3);
v = 110;
cout << v.capacity();
```

6. The C++ STL has many functions and functors. Here is your chance to try some of them. In a program when you use an STL algorithm add `#include<algorithm>`, and when you use an STL functor add `#include<functional>`.

   For many of these problems you will need to look up information online. Here are some sources:
   http://en.cppreference.com/w/cpp
   http://www.cplusplus.com/reference/algorithm/
   http://www.cplusplus.com/reference/std/functional/
   http://www.sgi.com/tech/stl/

   Fill in the correct code where you see a ****

```
vector<int> A {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
vector<int> B {1, 2, 1, 2, 1, 2};
vector<int> C{1, 2, 3, 1, 2, 3};
vector<int> D(6);
vector<int> E(10);
```

   (a) Copy the first 6 items of vector A into vector D

```
copy(****, ****, ****);
       // D now equals {1, 2, 3, 4, 5, 6}
```

   (b) Count the number of ones in vector B

```
cout << count(****, ****,  1);
       //prints out the number of times a one appears in B
```

   (c) In C++, there is a way to construct a unary functor from a binary functor. To do this you use an adapter, a function called `bind1st` or `bind2nd`. We use `bind1st` in this example to convert the STL binary predicate functor not_equal_to into a unary predicate by setting its first value to 1.

   Count the number of items that are not equal to one in B

```
cout << count_if(B.begin( ), ****, bind1st(not_equal_to<int>( ), 1));
       /*prints out the number of times a one doesn't appear in B.*/
```

   (d) Find the first item in vector A which equals 5

```
vector<int>::iterator vecItr;
vecItr = find(****, ****, 5);
        // returns an iterator to 5 in vector A
if (vecItr != A.end( ))
  cout << ****;
         // prints out the value pointed to by vecItr
```

(e) Find the first item in C that is greater than 2

```
vecItr = find_if(****, ****, bind2nd(greater<int>(), 2));
       // returns an iterator to 3 in C
if (vecItr != C.end( ))
  cout << ****;
         // prints out the value pointed to by vecItr
```

(f) Reverse the order of vector C

```
reverse(****, ****);
       // C is now reversed, it contains {3, 2, 1, 3, 2, 1}
```

(g) Sort the first 4 items in vector B

```
sort(B.begin( ), ****);
       // B now contains {1, 1, 2, 2, 1, 2}
```

Sort the the vector A in reverse order using the functor greater

```
sort(A.begin( ), ****, ****);
       //A now contains {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
```

(h) Choose any STL function and any STL functor that was not used previously in this assignment and use them on vector A.