CSc 103 Midterm Exam

Instructions

Read each question carefully. Calculators, books, phones, computers, notes, etc. are not permitted. All you need is something to write with. Scratch paper is provided at the front of the room. You'll have 60 minutes for the exam. **NOTE:** Don't use library functions that trivialize problems. See page 6 for a list of functions that are OK to use (or if you need a reminder on something), and be sure to ask before using anything that isn't on the list.

Name: ______________________________

Problem	Score
1 (10 points)	
2 (10 points)	
3 (10 points)	
4 (extra credit) (5 points)	
Total: 30 points	

e.g. int main() { ...
cin >> .. }

1. (10 points) Write a program that reads integers from standard input (see page 6 if you need a reminder on how to do that) and prints

   (a) the smallest even number, and
   (b) the largest odd number.

   For example, if the input were `10 3 2 9 1 7`, the output should be something like this:

```
smallest even: 2
largest odd:   9
```

   You can assume that all the standard `#includes` are there. Just write the `main()` function.

```
int main() {
    int se = INT_MAX;
    int lo = INT_MIN;
    int x; // next input.
    while(cin >> x) {
        if(x%2 == 0) {
            if(x < se)  se = x;
        } else { // odd
            if(x > lo)  lo = x;
        }
    }
    cout << 'smallest even' << se << '\n';
    cout << "largest odd' << lo << '\n';
    return 0;
}
```

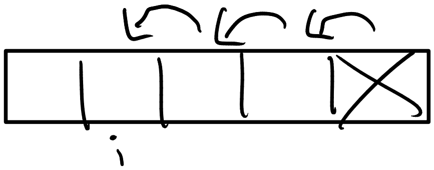
2. (10 points) Write a function `erase` that takes a vector (say of integers) and an **index** `i`, and removes the `i`-th element from the vector. (Yes, I know something like this exists in the standard library, but I want you to write it yourself for this problem.) An example to illustrate: if your vector `V` contained `1 2 3 4 5`, then after calling `erase(V,3)`, it should contain `1 2 3 5` (and the size of `V` should now be 4).

Here's a prototype:

```
void erase(vector<int>& V, size_t i);
```

IMPORTANT: Obviously you should NOT use the built-in `erase` function of the vector. This will not get you any credit.

Idea!



copy forward, starting @ index i, then shrink V by one.

```
void erase(vector<int>& V, sizet i) {
    for(sizet j = i; j < V.size()-1; j++) {
        V[j] = V[j+1];
    }
    V.pop_back();
    //or, V.resize(V.size()-1);

}
```

Not required!

```
int main() {
    vector<int> V = {1,2,3,4,5};
    erase(V, 3);
    // Now V = {1,2,3,5}

}
```

3. (10 points) Write a program that reads integers x in the range $0 \leq x < n$ from standard input, and then prints a frequency table of how many of each integer were entered. (You can assume n is a global constant in your program, but do not assume any particular value for it.) You don't have to do any error checking—if the program encounters any integers not in the range or any input that is not an integer, the behavior can be left undefined (it could just stop reading data and print the results it has so far, crash, or whatever). An example: if the program name is `ftable` and if $n = 8$ then executing

```
$ echo 0 0 7 3 1 3 3 7 6 | ./ftable
```

Need a count for each # from 0, ... n-1,

would produce output like this:

C
```
0: ~~0~~ ~~1~~ 2
1: ~~0~~ 1
2: 0
3: ~~0~~ ~~1~~ ~~2~~ 3
4: 0
5: 0
6: ~~0~~ 1
7: ~~0~~ ~~1~~ 2
```

option 1: store whole input as a vector, scan through over & over (n times)

You can assume that all the standard `#includes` are there. Just write the `main()` function. (Hint: use an array or a vector.)

option 2: use vector for the counts!

⊛ C[x] ≡ # times we've seen input x.

C =

0	0	0	0	0	0
0	1	2	3	4	5

```
int main() {
    vector<size_t> C;
    C.resize(n,0);  // now C[0,...n-1] are all 0.
    size_t x;
    while(cin >> x) {
        C[x]++;   // should check 0≤x<n ...
    }
    // now just print result:
    for(size_t i=0; i<n; i++){
        cout << i << ": " << C[i] << "\n";
    }
}
```

4. (5 points) Write a function that takes a vector of integers V and a target integer t, and returns a boolean value indicating whether or not there are two distinct elements of V such that $V[i] + V[j] = t$. **Important:** your algorithm should NOT perform a brute force search. If n is the size of V, it must take fewer than cn^2 steps asymptotically, for any value of c.

   Hint: begin by taking for granted a sorting algorithm that works in fewer than cn^2 steps. If you solve the problem using this mystery sorting algorithm, that will get you 4/5 points. Here's a prototype for such a sort function:

   ```
   void sort(vector<int>& V);
   ```

Cheat Sheet / Function Reference

Linked List Structure

```
struct node {
  int data;
  node* next;
  node(int d=0, node* n=NULL) : data(d),next(n) {}
};
```

Vectors

Use any of the following `vector` functions:

- `push_back(x)` (adds `x` to the end of the vector)
- `pop_back()` (removes whatever is at the end of the vector)
- `size()` (returns the number of elements)
- `resize(n)` and `resize(n,x)` (forces vector to have size `n`; if a second parameter `x` is given, any new values will be set to copies of `x`)
- `clear()` (remove all elements)
- `back()` (this gives the last element; `V.back()` is the same as `V[V.size()-1]`)

Strings

The `string` type actually supports **all** of the above vector functions, but you can also use the following:

- `x + y` (gives a new string that is the concatenation of strings `x` and `y`)
- `x += y` (appends string `y` to the end of `x`)
- `length()` (returns the number of characters; synonym for `size()`)

Constants

If you need the smallest thing that can be stored in an integer, it is `INT_MIN`; the largest is `INT_MAX`.

General

Also remember that you can access the elements of vectors and strings using the square brackets, e.g., `V[i]`, and that making an assignment between vectors or strings does what you expect (left hand side becomes a copy of the right hand side). You can also check vectors and strings for equality with `==` or other comparison operators like `<,>` (doesn't always make sense for vectors, though). And if you know about constructors, you can use those as well, but they will never be essential (might just shorten your solution a bit).