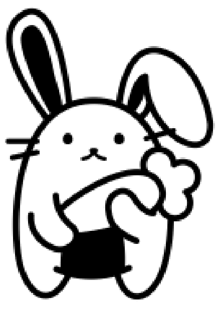
CSc 103 Midterm Exam

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) (10 points)	
Total: 30 points	



1. (10 points) Write a program that reads integers from standard input, and then prints the minimal value, the maximal value, and the average value. Note that the average may not be an integer. If the program encounters any input that is not an integer, it can just stop reading data and print the results it has so far. Here is an example: if the program name is `stats`, then executing

```
$ echo 8 99 3 32 14 83 0 11 53 | ./stats
```

would produce output like this:

```
min: 0
max: 99
average: 33.6666667
```

You can assume that all the standard `#includes` are there. Just write the `main()` function. (Hint: You don't even need vectors for this.)

```
int main()
{   int min = INT_MAX; // smallest so far
    int max = INT_MIN; // largest so far
    int x; // input
    int count = 0; // # of #s
    int sum = 0; // sum so far
    while(cin >> x) {
        if(x < min) min = x;
        if (x > max) max = x;
        count++;
        sum += x;
    }
    printf("min: %i\n max: %i\n average: %f\n", min, max,
                                (float) sum/count);
  return 0;
}
```

2. (10 points) Write a function that takes 2 vectors of **sorted** integers, and merges them into a third **sorted** vector which you can assume is initially empty. The prototype should look like this (note that the last reference parameter is for the *output*, which is why it is not `const`):

```
void merge(const vector<int>& V1, const vector<int>& V2, vector<int>& result);
```

As an example, if the first two vectors contain $[2, 4, 5, 9, 13, 17]$ and $[1, 3, 7, 8]$ respectively, then the result vector should contain $[1, 2, 3, 4, 5, 7, 8, 9, 13, 17]$. **Note: Do NOT use a sort function. You'll not receive any credit for doing the problem this way.**

```
{   size_t i1=0, i2=0;
    result.clear(); // optional
    while(i1 < V1.size() && i2 < V2.size()) {
        if (V1[i1] < V2[i2]) result.push_back(
                                      V1[i1++]);
        else result.push_back(V2[i2++]);
    }
    while(i1<V1.size()) result.push_back(V1[i1++]);
    while(i2<V2.size()) result.push_back(V2[i2++]);
}
```

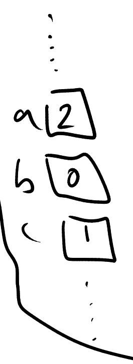
3. (10 points) Write a function that takes a string and returns another string consisting of all characters that appear *only once* in the input string. For example, if the input string was `"hello world"`, the returned string could be `" dehrw"` (the order isn't important, so `"hewrd "` would also be correct). *Hint:* perhaps use an array (or vector) of size 256, which notably would have an entry for every ASCII value... Here's a prototype:

```
string unduplicatedChars(const string& s){
    vector<size_t> C;
    C.resize(256, 0);   // all 256
                        // counters
                        //  = 0.
    for(size_t i=0; i<s.size(); i++)
        C[s[i]]++;
        // remember: characters are integers

    string u;
    for(int i=0; i<256; i++)
        if(C[i]==1)
            u += (char)i;

    return u;
}
```

Idea: use vector of counters, one for each character.



4. (10 points)

Write a function that takes two vectors of integers R, M and returns a single integer x such that the remainder of x when divided by $M[i]$ is exactly $R[i]$ for all $i = 0, \ldots, n-1$, where $n > 0$ is the size of both R and M.

Remarks.

- You can assume $R[i] \geq 0$ and $M[i] > 0$ for all i.
- Take for granted that the product of all elements of M will fit in a long unsigned integer:
$$\prod_{i=0}^{n-1} M[i] = M[0] \cdot M[1] \cdot \cdots \cdot M[n-1] < 2^{64}.$$
- If no such x exists, return the value -1.
- You can assume R, M have the same length, are non-empty.
- A brute force approach will get you 3 points. For 5 points, make something that works in fewer than $cn \log n$ steps, where n is the size of the vectors.

Cheat Sheet / Function Reference

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 at best they will just simplify your code a bit (non-default constructors will never be essential). Also, if you need to read all integers from `stdin`, remember you can just do this:

```
int x;
while (cin >> x) {
        /* do something with x... */
}
```