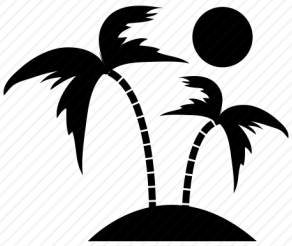

CSc 103 Final Exam

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 135 minutes for the exam. **NOTE:** Don't use library functions that trivialize problems. See page 7 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 (10 points)	
5 (extra credit) (5 points)	
Total: 40 points	

1. (10 points) Write a program that reads lines of text from standard input, and then prints the length and contents of the shortest and longest lines. In the case of a tie, any line of minimal/maximal length is OK to output. Example input/output follows. On input:

```
Hi Class!
I hope you are having fun with
this delightful exam!
:D :D :D
```

Output should look something like this:

```
Longest (30 chars): I hope you are having fun with
Shortest (8 chars): :D :D :D
```

You can assume that all the standard `#includes` are there. Just write the `main()` function. Remember: you can read lines from standard input using `getline` like this:

```
string line;
while (getline(cin,line)) {...}
```

2. (10 points) Write a function that takes a string by reference and reverses it. I want it to have the following prototype:

```
void reverse(string& s);
```

NOTE: If you have to allocate any new strings in your function, at most 8 points will be awarded. (I want you instead to just *rearrange the characters within the same string* that is input to the function.)

3. (10 points) Write a function that takes a linked list `L` (as a pointer to the first node) and a threshold value `t` and splits the list into two lists, `A` and `B`. List `A` should contain all nodes with `data < t`; list `B` will have everything else. Example: if `L = [2,42,33,1,7,72,17]`, then after calling `split(L,10,A,B)` you would have `A = [2,1,7]` and `B = [42,33,72,17]`.

   Important: It is OK if the order of resulting lists is different than what I show above. It is also OK if you destroy `L` in the process (reusing `L`'s nodes), or if you instead allocate new nodes for the result. Use the following prototype, and check page 7 if you need a reminder about the `node` structure:

```
/* NOTE: A and B are used for output! */
void split(node* L, int t, node*& A, node*& B);
```

4. (10 points) Consider the following recursive function:

```
vector<int> C(const vector<int>& P, const vector<int>& V, int n)
{
    if (n == 0) return V;
    vector<int> R;
    for (size_t i = 0; i < P.size(); i++)
        for (size_t j = 0; j < V.size(); j++)
            R.push_back(P[i]*V[j]);
    return C(P,R,n-1);
}
```

(a) What would be the output of the following?

```
int main()
{
    char T[2] = {'_','O'};
    vector<int> V = {1};
    vector<int> P = {1,0,1};
    vector<int> R = C(P,V,3);
    for (size_t i = 0; i < R.size(); i++)
        cout << T[R[i]];
    cout << "\n";
    return 0;
}
```

(b) In the following code segment, how many `1`'s would be in the vector `R` after the call to `C`?

```
vector<int> V = {1};
vector<int> P = {1,0,0,0,1};
vector<int> R = C(P,V,7);
```

5. (5 points) Suppose you have a possibly malformed linked list, which may contain a *cycle* (that is, the next field of a node may point back to a node already in the list). Write a function that would detect such a situation, with the following constraints:

   (a) Constant space overhead (irrespective of the length of the list or of the cycle, your program must allocate a **constant** amount of storage).

   (b) Linear time (the number of instructions that execute should have only linear dependence on the length of the list or the cycle).

   Failing to meet either of the criteria will result in no credit.

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]`)

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... */
}
```