COL 100 Re-Major Exam

Dec 6, 2018. 12:00 noon - 2:00 pm

Name:

Entry Number:

Group:

Notes:

- Total number of questions: 8. Max Marks: 40
- All answers should be written on the question paper itself.
- You can use any of the *Standard* or *Stanford* library functions while answering the questions.
- The last two sheets in the question paper are meant for rough work. If you run out of space, you can answer questions in this rough space. However, please clearly mention in the answer space for the appropriate question that we should look at the rough space for its answer.
- We will collect the question paper (including the last two sheets meant for rough work). We will not be collecting back any other rough sheets.

1. In the class, we have studied the Fibonacci series where each number in the series is the sum of last two numbers. Let us now consider a different kind of series. In this series, number at index i is sum of last two numbers if i is even, and it is the absolute of the difference between the last two numbers if i is odd. Let the 0^{th} and 1^{st} numbers in the series be 0 and 1, respectively. E.g., the first few numbers in the series would be $0, 1, 1, 0, 1, 1, 2, 1, 3...$ Write an iterative function to compute the n^{th} term of this series.

(Extra space for answering Q1)

2. Suppose you are given a file containing a newspaper article. Write a program which inputs this file and outputs a new file such that each line in the file is printed in the reverse order. For example, if the file is

```
COL100 is basic course in progamming
This is the last exam of the course
```

The output should be

```
programming in course basic is COL100
course the of exam last the is This
```

(Extra space for answering Q2)

3. Given a string s and a character ch, we would like to return the number of times ch appears in the string s. Write a recursive program (i.e., **no** loops) to achieve this functionality. Your program should be as efficient as possible. What is the time complexity of your program? Can you do this task in $O(n)$? Note that copying a string of size n takes $O(n)$ time. [5 points]

(Extra space for answering Q3)

4. Suppose you are given a vector v of integers. We say that a pair of numbers at indices i and j is out of order if $i < j$ and $v[i] > v[j]$. Write a function $countOfOrder(Vector<int> v)$ which takes as argument a vector v and returns the count of pairs which are out of order in v. Your program should be written an iterative manner, i.e., should not use recursion. What is the complexity of your program?

(Extra space for answering Q4)

5. Suppose you are given a vector of numbers. You would like to figure out if there is a subset of numbers in the vector which adds up to zero. Write a recursive function $checkAddsToZero(Vector<int> vec)$ which returns true if there is a subset of elements in vec which add to exactly zero, and false otherwise. E.g., if the vector is $\{1, 2, 3, 1, -6\}$ then the answer is true (since $1+2+3-6=0$). On the other hand if the vector is $\{1, 1, -11, 2, -5\}$, then the answer is false since there is no subset of elements which adds to zero.

(Extra space for answering Q5)

6. Given a vector of integers, you have to print all the subsets of the set of integers in the vector. Your implementation should be recursive. Example: if input =

```
{}
{1}
{1,2}
{1,2,4}
{1,4}
{2}
{2,4}
{4}
```

(Extra space for answering Q6)

7. Write a function $transpose(Vector<Vector> mat)$ which inputs a matrix as a vector of vectors, and computes its transpose. You can assume that input is a valid matrix.

(Extra space for answering Q7)

8. In Newton's method to find the roots of a function, we proceed as follows:

   - We start with an initial estimate: x
   - We compute the value of the function at $x : f(x)$.
   - We compute our new estimate as: $x = x - \frac{f(x)}{\nabla_x f(x)}$.
   - We repeat steps 3 and 4, until the value $|x| < e$.

   Here, $\nabla f(x)$ represents the derivative of f at value x. e is some user specified threshold. Write an iterative function *getQuadRoots*(*double* a, *double* b, *double* c, *double* x, *double* e) which takes the description of a quadratic function $f(x) = ax^2 + bx + c$ in the form of its coefficients as a input, another argument x as the starting point and a convergence threshold e, and returns the root found using Newton's method.
   (Extra space for answering Q8)