CSCE 221 Cover Page

Homework #1

Due February 11 at midnight to eCampus

E-mail address

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points

UIN

Last Name

First Name

User Name

or even zero, read more	e: Aggie Honor System O	office	
Type of sources			
People			
Web pages (provide URL)			
Printed material			
Other Sources			
I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work. "On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."			
Your Name		Date	

Type solutions to the homework problems listed below using preferably $\text{L}_{Y}X/\text{PT}_{E}X$ word processors, see the class webpage for more information about their installation and tutorials.

1. (10 points) Write the C++ classes called ArithmeticProgression and GeometricProgression that are derived from the abstract class Progression, with two pure virtual functions, getNext() and sum(), see the course textbook p. 87–90 for more details. Each subclass should implement these functions in order to generate elements of the sequences and their sums. Test your program for the different values of d, r and the number of elements n in each progression.

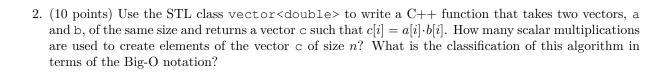
What is the classification of those functions: getNext() and sum() in terms of the Big-O notation? Recall the definitions of the arithmetic and geometric progressions.

 $\boldsymbol{Definition}$: An $arithmetic\ progression$ with the initial term a and the common real difference d is a sequence of the form

$$a, a+d, a+2d, \ldots, a+nd, \ldots$$

Definition: A geometric progression with the initial term a and the common real ratio r is a sequence of the form

$$a, ar, ar^2, \ldots, ar^n, \ldots$$



3. (10 points) Use the STL class vector<int> to write a C++ function that returns true if there are two elements of the vector for which their product is odd, and returns false otherwise. Provide a formula on the number of scalar multiplications in terms of n, the length of the vector, to solve the problem in the best and the worst cases. Describe the situations of getting the best and worst cases. What is the classification of the algorithm in the best and worst cases in terms of the Big-O notation?

4. (20 points) Write a templated C++ function called BinarySearch which searches for a target x of any numeric type T, and test it using a sorted vector of type T. Provide the formulae on the number of comparisons in terms of n, the length of the vector, when searching for a target in the best and the worst cases. Describe the situations of getting the best and worst cases. What is the classification of the algorithm in the best and worst cases in terms of the Big-O notation?

5. (10 points) (R-4.7 p. 185) The number of operations executed by algorithms A and B is $8n \log n$ and $2n^2$, respectively. Determine n_0 such that A is better than B for $n \ge n_0$.

6. (10 points) (R-4.21 p. 186) Bill has an algorithm, find2D, to find an element x in an $n \times n$ array A. The algorithm find2D iterates over the rows of A, and calls the algorithm arrayFind, see Code Fragment 4.5, p. 184, on each row, until x is found or it has searched all rows of A. What is the worst-case running time of find2D in terms of n? What is the worst-case running time of find2D in terms of N, where N is the total size of A? Would it be correct to say that find2D is a linear-time algorithm? Why or why not?

7. (10 points) (R-4.39 p. 188) Al and Bob are arguing about their algorithms. Al claims his $O(n \log n)$ -time method is always faster than Bob's $O(n^2)$ -time method. To settle the issue, they perform a set of experiments. To Al's dismay, they find that if n < 100, the $O(n^2)$ -time algorithm runs faster, and only when $n \ge 100$ is the $O(n \log n)$ -time algorithm better. Explain how this is possible.

8. (20 points) Find the running time functions for the algorithms below and write their classification using Big-O asymptotic notation. The running time function should provide a formula on the number of operations performed on the variable s. Note that array indices start from 0.

```
Algorithm E \times 1 (A):
   Input: An array A storing n \ge 1 integers.
  Output: The sum of the elements in A.
s \leftarrow A[0]
\quad \mathbf{for} \ i \leftarrow 1 \ \mathsf{to} \ n-1 \ \mathbf{do}
    s \leftarrow s + A[i]
end for
return s
Algorithm Ex2(A):
   Input: An array A storing n \ge 1 integers.
  Output: The sum of the elements at even positions in A.
for i \leftarrow 2 to n-1 by increments of 2 do
  s \leftarrow s + A[i]
end for
{f return} s
Algorithm Ex3(A):
    Input: An array A storing n \ge 1 integers.
    Output: The sum of the partial sums in A.
s \leftarrow 0
\quad \mathbf{for} \ i \leftarrow 0 \quad \mathbf{to} \ n-1 \ \mathbf{do}
    s \leftarrow s + A[0]
    for j \leftarrow 1 to i do
       s \leftarrow s + A[j]
    end for
end for
return s
Algorithm E \times 4 (A):
    Input: An array A storing n \ge 1 integers.
    Output: The sum of the partial sums in A.
t \leftarrow 0
s \leftarrow 0
for i \leftarrow 1 to n-1 do
    s \leftarrow s + A[i]
    t \leftarrow t + s
end for
return t
```