CSE 3131: ALGORITHM DESIGN 1

ASSIGNMENT (PART - A):

- *Assignment scores/markings depend on neatness and clarity.*
- *Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.*
- *You are allowed to use only those concepts which are covered in the lecture class till date.*
- *Plagiarized assignments will be given a zero mark.*

This assignment is designed to give you practice with:

- *Finding counter-examples*
- *Proofs of correctness*
- *Method of induction*
- *Asymptotic notations*
- *Recurrences*
- *Heap and Heap-sort*
- *Graph and related algorithms*
- *Greedy Method*
- *Divide-and-Conquer*

Counter Examples:

A natural place for counter examples to occur is when the converse of a known theorem comes into question. The **converse** of an assertion in the form "If P, Then Q" is the assertion "If Q, Then P". Whereas a complicated proof may be the only way to demonstrate the validity of a particular theorem, a single counter example is all that is need to refute the validity of a proposed theorem. For example, In Calculus you learn that if a function is differentiable at a point, then it is continuous at that point. What would the converse assert? It would say that if a function is continuous at a point, then it is differentiable at that point. But you know this is false. The counter example is $f(x) = |x|$. This function is continuous at $x = 0$, but it is not differentiable at $x=0$. This one counter example is all we need to refute the converse.

1. State the converse of "If a and b are even integers then a+b is an even integer". Show that the converse is not true by producing a counter example.

2. State the converse of "If a, b and c are real numbers such that $a + b = c$, then $(a+b)^2 = c^2$". Show that the converse is not true by producing a counter example.

3. State the converse of "If a, b and c are integers such that a divides b, then a divides the product bc." Show that the converse is not true by producing a counter example.

4. State the converse of "If a and b are rational numbers, then so is the product ab". Show that the converse is not true by producing a counter example.

Proofs of Correctness:

1. Consider the following algorithm:

```
Interchange (a,b)
//Input: Nonnegative integers a and b
//Output: a and b with swapped contents between them
1       a = a + b
2       b = a – b
3       a = a + b
```

Check the correctness of the pseudocode, using an experimental analysis (i.e. using counter examples).

2. Show that $a+b < \min(a,b)$. (Hint: use counter-examples)

3. Consider the following algorithm: (the lines starting with "//" are comments that will not be executed).

```
// precondition: a ϵ N, b ϵ N
m := a
```

```
    // loop invariant: $m \geq 0 \wedge \forall x \in N, a = bx + m$
    while $m \geq b$ do
        $m := m - b$
    end while
    // post-condition: $m = a \bmod b$, i.e., $0 \leq m < b \wedge \forall x \in N, a = bx + m$.
```

(a) Prove that the loop invariant is true before the first iteration of the loop.

(b) Let m denote the value of m at the end of some iteration of the loop and assume that the loop invariant is true for m. Furthermore, let x denote the value that makes a = bx + m true. Prove that the loop invariant remains true at the end of the next iteration of the loop, if there is such an iteration (use m to denote the value of m at the end of the next iteration).

(c) Prove that the post-condition is true after the last iteration of the loop. (You may assume that the loop invariant is true at the end of every iteration of the loop.)

4. Let A be a 1D array of n natural numbers, and consider the following pseudocode, with the precondition, post-condition and loop invariants inserted as comments:

```
//pre-condition: $\forall k \in N, 0 \leq k < n \rightarrow 0 \leq A[k]$
j =0
// Outer loop invariant: $\forall s \in N, 0 < j \leq s < n \rightarrow 0 \leq A[0] \leq A[1] \leq \ldots \leq A[j-1] \leq A[s]$
// Outer loop description: the first j elements are sorted in ascending order,
// and the jth element is less than or equal to any element between the j+1st and the nth element.
while(j < n){
    $i_{min}$ = j
    k = j +1
    // Inner loop invariant: $\forall r \in N, j \leq r < k \rightarrow A[i_{min}] \leq A[r]$
    // Inner loop description: finding smallest element between the j+1st and the nth element.
    while(k < n){
        if (A[k] < A[$i_{min}$]){
            $i_{min}$ = k
        }
        k ++
    } //end of inner loop
    temp = A[j]
    A[j] = A[$i_{min}$]
    A[$i_{min}$] = temp
    j ++
} //end of outer loop
//post-condition: $\forall j \in N, \forall k \in N, 0 \leq j < k < n \rightarrow A[j] \leq A[k]$
```

(a) Prove that the outer loop invariant is true at the start of the first iteration of the outer loop.
(b) Prove that if the outer loop invariant is true at the start of an iteration of the outer loop and the outer loop is executed, then the inner loop invariant is true at the start of the first iteration of the inner loop.
(c) Prove that if the inner loop invariant is true at the start of any iteration of the inner loop, then it is true at the end of that iteration.
(d) Prove that if the outer loop invariant is true when the outer loop terminates, then the post-condition is true.

5. Examine the method mult(m, n) below. Prove or disprove that if the loop invariant is true at the beginning of loop iteration, then it is true at the end of a loop iteration. Your proof should be in the structured proof format from class.
// mult returns the product of integers m and n
// precondition: m and n are integers
// post-condition: integer product mn is returned

```
int mult (int m, int n) {
    int x = m, y = n, z = 0;
    // loop condition: z = mn - xy
    while (x != 0) {
        if (x % 2 != 0) {
            if (x * m > 0) {
                z = z - y;
            }
            else {
                z = z + y;
            }
        }
        x = (int)(floor(x / -2.0));
        y = y * 2;
    }
    // post-condition z = mn
    return z;
    }
}
```

6. Prove that the following algorithm for exponentiation is correct.

```
*function* power(y, z)
// comment Return yz , where y ϵ R, z ϵ N
    x := 1;
    while z > 0 do
        x := x · y
        z := z − 1
    return(x)
```

7. Prove that the following algorithm for the multiplication of natural numbers is correct.

```
*function* multiply(y, z)
// comment Return yz, where y, z ϵ IN
    x := 0;
    while z > 0 do
        if z mod 2 = 1 then x := x + y;
    y := 2y; z := z/2 ;
    return(x)
```

8. Consider the following merge function. Is it correct?

INPUT: A[1..n1], B[1..n2] sorted arrays of integers
OUTPUT: permutation C of A.B s.t. C[1] ≤ C[2] ≤ ... ≤ C[n1+n2]

```
    i = 1
    j = 1
    for k = 1 to n1 + n2 do
        if A[i] <= B[j] then
            C[k] = A[i]
            i++
        else
            C[k] = B[i]
            j++
    return C
```

Method of induction:

1. Prove that: for $n \geq 5$, $4n < 2^n$.

2. Prove that: for all $n > 0$, $n^3 \leq n^2$

Asymptotic notations and Complexity analysis:

1. Here is a bunch of functions of one variable, n.
   $\sqrt{}$ n, $2^n - 20n$, $n^2 - 20n$, n lg n , $n^2 - n^3 + n^4$, n^2 lg n, $(\lg n)^2$, 17n
   Place them in a list from left to right, so that if f (n) is any function in your list and g(n) any function to its right, we have f (n) = O(g(n)). You do not need to justify your answer.
   (For instance, if the functions were 5n, 23, n^2 , the correct answer would be 23, 5n, n^2 , since 23 = O(5n) and 5n = O(n^2).)

(In this course, lg n represents the binary logarithm of n, i.e., $\lg n = \log_2 n$.)

2. (a) Prove or disprove: If $f(n) = O(g(n))$, then $\log(f(n)) = O(\log(g(n)))$.
   (b) Prove or disprove: If $f(n) = O(g(n))$, then $2^{f(n)} = O(2^{g(n)})$.
   (c) Prove that $\log^k n = o(n^{1/k})$ for any positive integer k.

3. Consider the following code fragment.

```
for i =1 to n do
        for j = i to 2*i do
                output "ALGORITHMS"
```

Let T(n) denote the number of times "ALGORITHMS" is printed as a function of n.

a) Express T(n) as a summation (actually two nested summations)

b) Simplify the summation. Show your work.

4. Consider the following algorithm:

```
Sum(n)
// Input: A nonnegative integer n
S ← 0
for i ← 1 to n do
        S ← S + i
return S
```

(a) What does this algorithm compute?

(b) What is its basic operation?

(c) How many times is the operation executed?

(d) What is the efficiency of this algorithm?

(e) Suggest an improved algorithm and indicate its efficiency.

Recurrences:

Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function f(n). You do not need to turn in proofs (in fact, please don't turn in proofs), but you should do them anyway just for practice. Assume reasonable but nontrivial base cases if none are supplied. More exact solutions are better.

1. $A(n) = A(n/3 + 5 + \log n) + n \log \log n$
2. $B(n) = \min_{0<k<n}(3 + B(k) + B(n - k))$.
3. $D(n) = n/n{-}5\ D(n - 1) + 1$
4. $E(n) = E(3n/4) + 1/\sqrt{n}$
5. $F(n) = F(\log_2 n) + \log n$
6. $G(n) = n + 7\sqrt{n} \cdot G(\sqrt{n})$
7. $H(n) = \log 2\ (H(n - 1)) + 1$
8. $I(n) = I(n^{1/4}) + 1$
9. $J(n) = J(n - \lfloor n/\log n \rfloor) + 1$

HEAP and HEAPSORT:

1. Is the array with values <23, 17, 14, 6, 13, 10, 1, 5, 7, 12> a max-heap? If not then build the max heap.

2. Build the min heap of Q1.

3. Illustrate the operation of MAX-HEAPIFY (A,3) on the array A =<27, 17,3,16,13,10,1,5,7,12,4,8,9,0>.

4. Illustrate the operation of HEAPSORT on the array A=<5,13,2,25,7,17,20,8, 4>.

GRAPH:

1. Write the code and algorithm for following algorithm:

BFS, DFS, PRIM'S MST, KRUSKAL's MST, DIJKSTRA's Shortest Path.

2. For detecting the cycle in a graph what algorithm you will use and how. Describe it properly.

3. There are n cities and there are roads in between some of the cities. Somehow all the roads are damaged simultaneously. We have to repair the roads to connect the cities again. There is a fixed cost to repair a particular road. Find out the minimum cost to connect all the cities by repairing roads. Input is in matrix(city) form, if city[i][j] = 0 then there is no road between city i and city j, if city[i][j] = a > 0 then the cost to rebuild the path between city i and city j is 'a' . Find the minimum cost to connect all the cities. It is sure that all the cities were connected before the roads were damaged.

GREEDY METHOD:

Write the C/C++/Python/Java code for the following problems using Greedy method:

1. Interval Scheduling
2. Interval Partitioning
3. Optimal Interval Scheduling by Minimizing Lateness
4. Optimal Caching
5. Huffman Tree Construction

DIVIDE-and-CONQUER:

Q1. Convert the coefficient representation of a polynomial function into the point representation using Divide and Conquer in O(nlogn) time.

Q2. Multiply two integer number X=345 and Y=45 in O(nlogn) using FFT and Convolution. Hints: Convert X and Y into two polynomial function, such that P(x)=5+4x+3x2 and Q(x)=5+4x. When x=10. Now use the concept of (Q1).

Write the C/C++/Python/Java code for the following problems using Divide-and-Conquer method:

1. Binary Search
2. Merge Sort
3. Counting Inversion
4. Quick Sort using (i) Lomuto's Partition, (ii) Randomized Partition
5. Integer Multiplication (Karatsuba algorithm)
6. Closest Pair of Points

ASSIGNMENT (PART - B): - PROJECT

Submit an algorithm-based project report with the following details:

- Project title
- Input specifications
- Conditions or constraints, if any
- Goal/Objective/Desired output
- Brief description

- Mathematical formulation with model
- Algorithm design paradigm used
- Pseudocode
- Code
- Results analysis
- Real-life applications
- Statement of declaration for genuinity of the project

Some example scenarios for project:

- Dynamic Road Traffic Management based on Kruskal's Algorithm
- Implementation of Dijkstra's algorithm to find an effective route to avoid traffic jam on a busy hour
- Traffic And Alternate Route Display Using Dijkstra's Algorithm
- Vehicle Text Data Compression and Transmission Method Based on Optimized Huffman Encoding Algorithms

Submission Process:

For Part-A: Scan each page of your hand-written assignment, then convert each scanned page to .pdf and then merge pdf of each page to a single pdf which is to be finally sent.

For Part-B: Send a zip folder (name of the zip folder must be your registration number_AD1) containing the project report, code and output file/screen-shot of each algorithm mentioned. On the top of each program, you must mention your full name, registration number, title of the program and date.

Finally in a single mail, send the .pdf of Part-A and zip folder of PartB to your respective AD1 class teacher by the due date, i.e. 16.07.2021 (Friday). Subject of the mail should be "full name_registration number_AD1 Assignment".