Honors Computer Science I (COP 3502) Exam #2
Date: 3/26/2026

Name: _____

1) (8 pts) Show the state of the array below after each iteration of Insertion Sort:

16	18	13	2	11	27	14	5	22	9
2	5	9	11	13	14	16	18	22	27

2) (4 pts) Why does Quick Sort, on average, run faster than Merge Sort, in practice?

3) (6 pts) Consider running Merge Sort on the array below (of size 8). Over the course of the algorithm, 7 Merge operations occur. Show the state of the array after each of the first six merge operations:

Array	13	16	18	4	11	1	19	5
1st Merge								
2nd Merge								
3rd Merge								
4th Merge								
5th Merge								
6th Merge								
7th Merge	1	4	5	11	13	16	18	19

4) (9 pts) Write down the solutions to each of these recurrence relations (Big-Oh bounds) using the Master Theorem:

(a) $T(n) = 4T\left(\frac{n}{2}\right) + O(n^3)$ _____

(b) $T(n) = 8T\left(\frac{n}{4}\right) + O(n)$ _____

(c) $T(n) = 9T\left(\frac{n}{3}\right) + O(n^2)$ _____

5) (12 pts) Solve the following recurrence relation using the iteration technique shown in class. Please obtain a Big-Oh bound in terms of n for the recurrence relation.

$$T(n) = 8T\left(\frac{n}{2}\right) + O(n^2), \text{ for n} > 1$$
$$T(1) = 1$$

6) (12 pts) A common function to run on a binary search tree is the lower function, which returns a pointer to the node storing the largest value in a binary search tree strictly less than a given input value. If no such value exists, then the function should return NULL. Write an implementation of this function using the struct shown below:

```
typedef struct bstnode {
    int data;
    struct bstnode* left;
    struct bstnode* right;
} bstnode;

bstnode* lower(bstnode* root, int key) {




}
```

7) (20 pts) Consider the following problem: given a list of integers, imagine inserting either plus signs or times signs between each integer, how many different values could the possible expression form? We can solve this problem by trying each possible combination of signs, and for each combination of signs, evaluating the expression, and marking, in a boolean array, each value produced. Then, we could just count the number of entries equal to 1 in the boolean array. On the next 2 pages is a solution to the problem which runs a small test case with two functions left empty for you to complete. Complete those functions.

```
#include <stdio.h>
#include <stdlib.h>
#define N 5

int total(int* possible, int len);
void go(int* signs, int k, int* values, int* possible, int n);
int eval(int* signs, int* values, int* possible, int n);

int main() {
    int values[N+1] = {2,3,1,2,4,3};
    int prod  = 1;
    for (int i=0; i<=N; i++) prod *= values[i];
    int max = prod + N + 1;
    int* possible = calloc(max, sizeof(int));
    int* signs = calloc(N, sizeof(int));
    go(signs, 0, values, possible, N);
    printf("possible = %d\n", total(possible, max));
    free(possible);
    free(signs);
    return 0;
}

// Returns the number of values in the array possible equal to 1.
// The length of the array possible is len.
int total(int* possible, int len) {









}

// Fills in the possible array with all possible values that can be
// formed given that the first k signs are fixed. signs[i] = 0 indicates
// that the ith sign is an addition sign. signs[i] = 1 indicates that
// the ith sign is a multiplication sign.
void go(int* signs, int k, int* values, int* possible, int n) {









}
```

```
int eval(int* signs, int* values, int* possible, int n) {

    int res = 0, curProd = values[0];
    for (int i=0; i<n; i++) {
        if (signs[i] == 0)
            res += curProd;
            curProd = values[i+1];
        }
        else {
            curProd *= values[i+1];
        }
    }

    return res + curProd;
}
```

8) (8 pts) Evaluate the following postfix expression, showing the contents of the stack as the expression is being evaluated at each of the different points shown (A, B and C) in the expression:

						A			B		C					
3	5	+	2	6	3	/	*	2	3	+	*	4	/	–		

Stack at A

Stack at B

Stack at C

Value of Expression = _____

9) (20 pts) Consider using a linked list to store big integers, digit by digit. The linked list stores the digits in reverse order. For example, 1847 would be stored in the list → 7 → 4 → 8 → 1. The reason for this storage is it makes most mathematical operations easier to compute. Given the struct definition below, write an add method that takes in pointers to two nodes, and adds the two numbers represented storing the new number as a newly allocated linked list of nodes and returns a pointer to the list storing the corresponding sum. **Note: since no carry value is passed into the function, you have to solve it iteratively.**

```
typedef struct node {
    int digit;
    struct node* next;
} node;

node* add(node* op1, node* op2) {









}
```

10) (1 pt) The Miami Open, a tennis tournament, is currently being held at the grounds of Hard Rock Stadium. What company owns the naming rights to the stadium?

_____

C Language Reference

```
// Allocates size bytes of uninitialized storage.
void* malloc(size_t size);

// Allocates a block of memory for an array of num elements,
// each of them size bytes long, and initializes all of its
// bits to 0.
void* calloc(size_t num, size_t size);

// Reallocates the given area of memory. It must be previously
// allocated by malloc, calloc or realloc and not yet freed
// with a call to free or realloc.
// This is done by either expanding or contracting the existing
// area pointed to by ptr, if possible. If not, a new memory
// block of size new_size bytes is allocated, copying memory
// area with size equal the lesser of the new and the old sizes,
// and freeing the old block. If there is not enough memory,
// the old memory block is not freed and null is returned.
void* realloc(void* ptr, size_t new_size);
```

Scratch Work: Please clearly mark any work below you would like graded.