# Computer Science Foundation Exam

## January 17, 2026

## Section A

## BASIC DATA STRUCTURES

**NO books, notes, or calculators may be used,**
**and you must work entirely on your own.**

**PLEASE USE CAPITAL LETTERS IN WRITING YOUR NAME**

**Last Name:** _____

**First Name:** _____

**UCFID:** _____

| Question # | Max Pts | Category | Score |
|---|---|---|---|
| 1 | 10 | DSN | |
| 2 | 5 | ALG | |
| 3 | 10 | ALG | |
| TOTAL | 25 | ---- | |

**You must do all 3 problems in this section of the exam.**

**Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.**

**1)** (10 pts) DSN (Dynamic Memory Management in C)

In the Mario Kart video game series, drivers collect items (such as shells, bananas, and mushrooms) during a race. Each driver maintains a list of collected items that may grow as new items are obtained. Given the following typedef structure definitions:

```
//struct representing an item
typedef struct {
    char itemName[15];
} item_t;

//struct representing a driver
typedef struct {
    char driverName[20];
    item_t *items;
    int itemCount;
} driver_t;
```

complete the following function `addItem`. This function will store a new item that a driver collects during a race. In particular, the string inside `newItem` needs to be copied into the newly allocated memory within `driver`. The function returns 1 if the item was successfully added. Otherwise, 0 is returned. The following assumptions can be made:

- The driver can potentially have no items initially, which would be represented by the value 0 in the component `itemCount`.
- You may assume that all items inserted fit within the array size of 15 elements. There is no need to do a conditional check.
- A driver can hold at most 3 items. (addItem should return 0 if the driver already has 3 items.)

```
int addItem(driver_t *driver, const item_t *newItem) {




}
```

**2)** (5 pts) ALG (Linked Lists)

Suppose we have a singly linked list implemented with the structure below and a function that takes in the head of the list and an integer.

```
typedef struct node_s{
     int val;
     struct node_s * next;
}node_t;

node_t *mystery(node_t *head) {

    node_t *second, *rest, *tail;

    if(head == NULL || head->next == NULL)
        return head;

    second = head->next;
    rest = mystery(second->next);
    head->next = rest;
    tail = head;

    while(tail->next != NULL)
        tail = tail->next;

    tail->next = second;
    second->next = NULL;

    return head;
}
```

If we call head = mystery(head); on the following list, show the list after the function has finished.

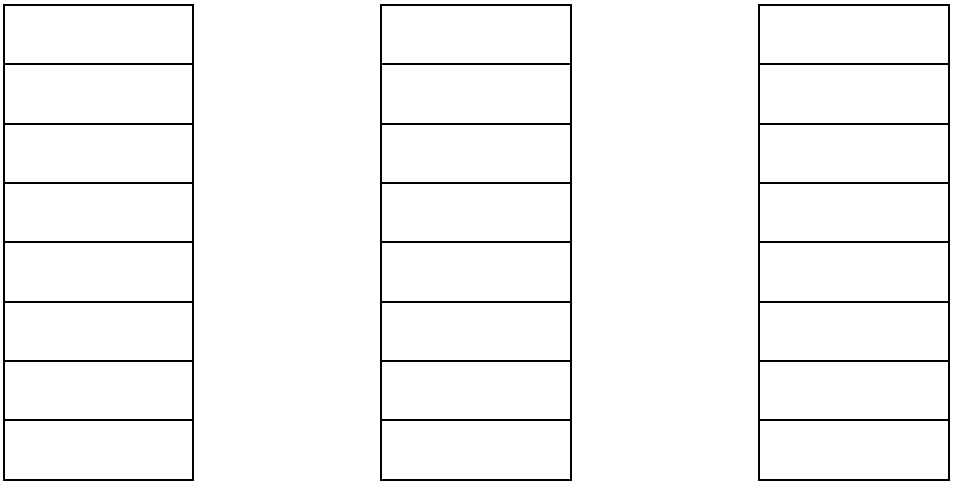
head → 1 → 2 → 3 → 4 → 5? Please fill in the designated slots below. (Note: The list does have five items in it after the function call executes.)

head → _____ → _____ → _____ → _____ → _____

**3)** (10 pts) ALG (Stack)

Convert the following infix expression to postfix using a stack. Show the contents of the stack at the indicated points (A, B, and C) in the infix expression.

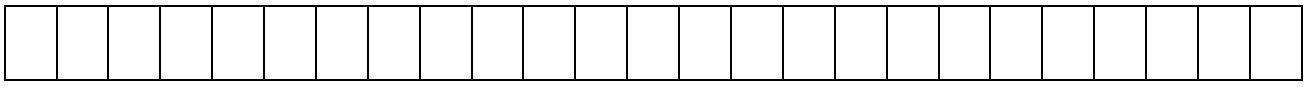
```
                A                   B                   C
7 * (3 + 9) - 4 / 6 + (8 * 2 - (5 + 7)) / 3 - 9 * 4 + 6
```



A B C

Note: A indicates the location in the expression **AFTER** the minus operator and before the value 4. B indicates the location in the expression **AFTER** the value 5 and before the plus operator. C indicates the location in the expression **AFTER** the value 4 and before the plus operator. Please use the cell closest to the marked letter (A, B or C) as the bottom of the stack.

Resulting postfix expression:



**Note: There are exactly the correct number of boxes above. These should be filled with 13 numbers and 12 operators.**

# Computer Science Foundation Exam

## January 17, 2026

## Section B

## ADVANCED DATA STRUCTURES

**NO books, notes, or calculators may be used,**
**and you must work entirely on your own.**

**PLEASE USE CAPITAL LETTERS IN WRITING YOUR NAME**

**Last Name:** _____

**First Name:** _____

**UCFID:** _____

| Question # | Max Pts | Category | Score |
|---|---|---|---|
| 1 | 10 | ALG | |
| 2 | 5 | DSN | |
| 3 | 10 | ALG | |
| TOTAL | 25 | ---- | |

**You must do all 3 problems in this section of the exam.**

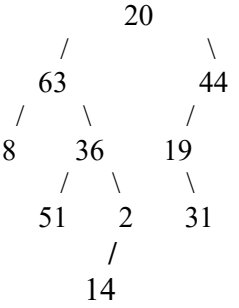
**Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.**

**1)** (10 pts) ALG (Binary Trees)

Consider the following binary search tree and the `printMystery` function shown below. What would be printed by the function if we pass the root of the following tree? Please place each of the ten numbers that get printed in the order they get printed in the blanks provided at the bottom of the page.

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

void printMystery(treenode* root) {

    if (root == NULL)
        return;

    printMystery(root->right);
    printf("%d ", root->data);
    printMystery(root->left);
}
```

Tree:




_____ _____ _____ _____ _____ _____ _____ _____ _____ _____

**2)** (5 pts) DSN (Hash Tables)

Consider the following hash function for a string, s, of lowercase letters, where value('a') = 1, value('b') = 2, ..., value('z') = 26, and the length of the string is n.

f(s, m) = ( value(s[0]) x $27^0$ + value(s[1]) x $27^1$ + value(s[2]) x $27^2$ + ... value(s[n-1]) x $27^{n-1}$ ) mod m.

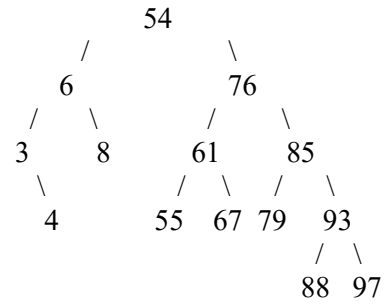
Complete the function below so that it computes this hash function. **Do not call the pow function.** (Any solution with a call to the pow function will get an automatic 0.) Remember all computations must occur "under mod." You may assume that the value of m is small enough that if coded appropriately no overflow errors will occur.

```
int f(char* s, int m) {

    int res = 0;
    int pow27 = 1;
    int len = strlen(s);

    for (int i=0; i<len; i++) {

        // Update res to equal the running value of the hash function
        // so far.

        ____________________________________________________ ;

        // Update pow27 to be the current power of 27 under mod.

        ____________________________________________________ ;

    }

    return res;
}
```

**3)** (10 pts) ALG (AVL Trees)

Consider the following AVL tree. Delete 8 from the tree and show the final resulting AVL tree. In the process of the delete, the tree gets restructured twice. Draw a box around the full tree at the following stages of the process:

(a) (5 pts) Right after the first restructuring takes place.
(b) (5 pts) At the end of the process, right after the second restructuring takes place.



# Computer Science Foundation Exam

## January 17, 2026

## Section C

## ALGORITHM ANALYSIS

**NO books, notes, or calculators may be used,**
**and you must work entirely on your own.**

**PLEASE USE CAPITAL LETTERS IN WRITING YOUR NAME**

**Last Name:** _____

**First Name:** _____

**UCFID:** _____

| Question # | Max Pts | Category | Score |
|---|---|---|---|
| 1 | 5 | ANL | |
| 2 | 10 | ANL | |
| 3 | 10 | ANL | |
| TOTAL | 25 | ---- | |

**You must do all 3 problems in this section of the exam.**

**Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.**

**1)** (5 pts) ANL (Algorithm Analysis)

Let T(n, m) be the run-time of the following function, in terms of input parameters n and m. **Write down a recurrence relation that T(n, m) satisfies.**

```
int f(int* values, int n, int m) {

    if (n == 1) return values[0];

    int left = f(values, n/2, m-1);
    int right = f(values+n/2, n/2, m-1);

    while (m > 0) {
        if (left > right)
            left--;
        else
            right++;
        m--;
    }

    return 2*(left+right);
}
```

**2)** (10 pts) ANL (Algorithm Analysis)

An algorithm which has a run time of $O(n^2\sqrt{n})$ takes 3 seconds to run on an input with size n = 10,000. (Note: the function in the Big-Oh is read out loud as, "n squared times square root n.") If there are 86,400 seconds in a day, how many days would the algorithm take to complete on an input size of $n = 10^6$? **Express your answer as a fraction in lowest terms.** Put a box around your final answer.

**3)** (10 pts) ANL (Recurrence Relations)

Determine a closed form solution to the following recurrence relation, in terms of n. (Your solution must be an exact function in terms of n, not a Big-Oh bound.)

$$T(n) = 3T(n-1) + 3^n, \text{ for integers n > 1}$$
$$T(1) = 12$$

# Computer Science Foundation Exam

## January 17, 2026

## Section D

## ALGORITHMS

**NO books, notes, or calculators may be used,**
**and you must work entirely on your own.**

**PLEASE USE CAPITAL LETTERS IN WRITING YOUR NAME**

**Last Name:** _____

**First Name:** _____

**UCFID:** _____

| Question # | Max Pts | Category | Score |
|---|---|---|---|
| 1 | 10 | DSN | |
| 2 | 10 | DSN | |
| 3 | 5 | ALG | |
| TOTAL | 25 | ---- | |

**You must do all 3 problems in this section of the exam.**

**Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.**

**1**(10 pts) DSN (Recursive Coding)

Consider the problem of cutting a chocolate bar that is **length** inches long and **width** inches wide into multiple 1 inch by 1 inch squares. You may either cut the bar horizontally or vertically to create two rectangular bars that are an integer number of inches in length and width and are as close to equal size as possible. For example, a bar that is 8 inches long and 3 inches wide can be cut into two bars that are both 4 inches long and 3 inches wide, OR into two bars where one is 8 inches long and 1 inch wide and the other bar is 8 inches long and 2 inches wide. From there, both bars must be recursively cut. Note that if one dimension is a single inch, only a single cut is possible (reducing the larger dimension.) The cost of a cut that leaves the length unchanged is equal to **length** and a cut that keeps the width unchanged is equal to **2*width**. Write a recursive function that takes in the parameters **length** and **width** (both positive integers), and returns the minimum cost of cutting a chocolate bar with those dimensions into 1inch by 1 inch squares.

```
int minCutCost(int length, int width) {

}
```

**2)** (10 pts) DSN (Sorting)

Although the example code traditionally shown for the Bubble Sort is iterative, the algorithm itself lends itself easily to recursion. (After one pass of Bubble Sort on an array of size n, the work that remains is a problem of the exact same nature.) Write a **recursive** implementation of the Bubble Sort algorithm that sorts elements in ascending order in the function shown below.

```
void bubbleSortRec(int* array, int n) {



}
```

**3)** (5 pts) ALG (Base Conversion)

Convert each of the following binary numbers into base 16 (Hexadecimal). No need to show your work, credit will be based solely on the answers.

(a) `10010110`

_____

(b) `11001111`

_____

(c) `101101`

_____

(d) `10111101011`

_____

(e) `1001101101000010000`

_____