# Engineering C Exam Study Plan

## Day 1: Sunday - Fundamentals & Control Flow

**Focus:** Variables, operators, conditionals, loops **Time:** 3-4 hours

### Morning Session (1.5 hours)

1. **Enhanced Resistor Calculator**
   - Create a program that reads 3 resistor values
   - Calculate all possible series/parallel combinations
   - Handle edge cases (0 or negative values)
   - Display results sorted by total resistance
2. **Advanced Number Analyzer**
   - Read integers until -999 is entered
   - Calculate: sum, average, min, max, count of evens/odds
   - Find the second largest number
   - Display prime numbers from the input
3. **Pattern Generator** Write programs to generate these patterns (user inputs size):

```
Pattern A:        Pattern B:        Pattern C:
1                 1 2 3 4 5         1
1 2               2 3 4 5 6         2 1
1 2 3             3 4 5 6 7         3 2 1
1 2 3 4           4 5 6 7 8         4 3 2 1
1 2 3 4 5         5 6 7 8 9         5 4 3 2 1
```

### Afternoon Session (1.5 hours)

4. **Calculator with Memory**
   - Implement +, -, *, /, % operations
   - Add memory functions (MS, MR, MC, M+)
   - Include power and square root
   - Handle division by zero gracefully
5. **Number Base Converter**
   - Convert decimal to binary, octal, hex (without printf formatting)
   - Use loops and arrays to store digits
   - Display step-by-step conversion process

## Day 2: Monday - Functions, Arrays & Strings

**Focus:** Function design, arrays, string manipulation **Time:** 4 hours

### Morning Session (2 hours)

1. **Statistics Library** Create functions for:
   - `float harmonic_mean(float arr[], int size)`
   - `float geometric_mean(float arr[], int size)`
   - `void remove_outliers(float arr[], int *size, float threshold)`
   - `int find_mode(int arr[], int size)`
2. **String Processor** Implement these functions:
   - `void reverse_words(char str[])` - reverse each word in place
   - `int count_palindrome_words(char str[])`
   - `void encode_rot13(char str[])`
   - `int validate_email(char str[])`
3. **Matrix Operations**
   - Read two 3x3 matrices
   - Implement matrix multiplication
   - Calculate determinant
   - Find transpose and check if symmetric

### Afternoon Session (2 hours)

4. **Text Analysis Tool**
   - Read multiple lines until "END"
   - Count: sentences, words, average word length
   - Find most frequent word
   - Generate readability score
5. **Tic-Tac-Toe Game**
   - Use 2D array for board
   - Functions for display, move validation, win checking
   - Implement AI opponent (simple strategy)

## Day 3: Tuesday - Pointers, Files & Structures

**Focus:** Advanced topics, memory management **Time:** 4-5 hours

## Morning Session (2.5 hours)

1. **Pointer Gymnastics**
   - Implement `void swap_arrays(int *arr1, int *arr2, int size)`
   - Create `char* find_substring(char *str, char *sub)` without string.h
   - Write `void reverse_array(int *arr, int size)` using only pointers
   - Implement your own `memcpy` function
2. **Student Database System**

```c
typedef struct {
    char name[50];
    int id;
    float grades[5];
    struct Date {
        int day, month, year;
    } enrollment_date;
} Student;
```

   - File operations: save/load student records
   - Sort by name, ID, or GPA
   - Search functionality
   - Calculate class statistics
3. **Dynamic Memory Challenges**
   - Implement a growable array (like vector)
   - Create a string builder that concatenates efficiently
   - Memory pool allocator (pre-allocate chunks)

## Afternoon Session (2 hours)

4. **File Encryption Tool**
   - Read binary file
   - XOR encryption with password
   - Save encrypted/decrypted file
   - Add checksum verification
5. **CSV Data Processor**
   - Read CSV with unknown columns/rows
   - Dynamic memory allocation for data
   - Sort by any column

- Export filtered results

## Day 4: Wednesday - Advanced Topics & Mixed Practice

**Focus:** Linked lists, sorting, comprehensive review **Time:** 5 hours

### Morning Session (2.5 hours)

1. **Enhanced Linked List** Implement a doubly-linked list with:
   - Insert sorted
   - Remove duplicates
   - Reverse list
   - Merge two sorted lists
   - Find nth node from end
2. **Sorting Algorithm Comparison**
   - Implement bubble, selection, insertion, and merge sort
   - Time each algorithm on various data sizes
   - Visualize sorting steps
   - Handle different data types using function pointers
3. **Expression Evaluator**
   - Parse and evaluate mathematical expressions
   - Support +, -, *, /, parentheses
   - Use stack (implemented as linked list)
   - Handle errors gracefully

### Afternoon Session (2.5 hours)

4. **Mini Banking System** Combine everything learned:
   - Account structures with transactions
   - File persistence
   - Linked list for transaction history
   - Function pointers for operation dispatch
   - Dynamic memory for accounts
   - Menu-driven interface
5. **Code Golf Challenges** Solve these in minimal code:
   - FizzBuzz with twist (custom rules)
   - Pascal's triangle generator
   - Maze solver (2D array)

- Prime factorization with display

## Final Evening Review (Wednesday Night)

**Time:** 2 hours

### Quick Fire Round (30 min)

- Write 5 different swap functions (by value, reference, XOR, etc.)
- Implement strlen, strcpy, strcat without string.h
- Quick sorting of 5 elements by hand (trace bubble & selection)

### Common Pitfalls Review (30 min)

- Array bounds and off-by-one errors
- Pointer arithmetic gotchas
- Memory leaks in linked lists
- File handling errors
- Integer division truncation

### Exam Strategy Practice (1 hour)

1. **Time Management Exercise**
   - Set timer for 15 minutes per question
   - Practice explaining code modifications
   - Focus on clean, commented code
2. **Error Debugging** I'll give you buggy code to fix:
   - Segmentation faults
   - Logic errors
   - Memory leaks
   - Compilation errors

## Daily Tips:

- **Compile frequently** with `-Wall -Wextra`
- **Test edge cases**: empty input, single element, maximum size
- **Comment your code** as you write
- **Use meaningful variable names**
- **Check return values** from malloc, fopen, scanf

## Exam Day Checklist:

- [ ] Review pointer syntax
- [ ] Remember string null terminators
- [ ] Check array bounds
- [ ] Free all malloc'd memory
- [ ] Close all opened files
- [ ] Handle error cases
- [ ] Use proper formatting for output

Remember: The exam tests both your code writing AND your ability to explain/modify code. Practice talking through your solutions!