Mock Exam 2: Lectures 7-12 (Strings, Command Line, Pointers, Dynamic Memory, Multi-file Projects)

Time Limit: 1 hour
Total Marks: 100

Section A: Multiple Choice Questions (32 marks)

Choose the best answer for each question. 2 marks each.

Question 1: How is a string terminated in C? a) With a space character b) With the null character '\0' c) With a newline character d) Automatically by the compiler

Question 2: What will `strlen("Hello")` return? a) 6 b) 5 c) 4 d) Error

Question 3: In `main(int argc, char *argv[])`, what does `argc` represent? a) The first argument b) The number of command line arguments c) The program name d) An array of arguments

Question 4: What is `argv[0]` always equal to? a) The first user argument b) The program name c) The number of arguments d) NULL

Question 5: What does the `*` operator do with pointers? a) Gets the address of a variable b) Gets the value a pointer points to (dereference) c) Creates a new pointer d) Multiplies two values

Question 6: What does the `&` operator do? a) Gets the value of a variable b) Gets the address of a variable c) Creates a pointer d) Performs logical AND

Question 7: What will this code print?

```
int x = 42;
int *ptr = &x;
printf("%d", *ptr);
```

a) 42 b) The address of x c) 0 d) Error

Question 8: What does `malloc()` do? a) Allocates memory on the stack b) Allocates memory on the heap c) Frees memory d) Creates a pointer

Question 9: What must you do after using `malloc()`? a) Nothing b) Call `free()` c) Call `delete()` d) Set the pointer to 0

Question 10: Which function safely copies one string to another? a) `strcopy()` b) `strcpy()` c) `copy()` d) `stringcopy()`

Question 11: What happens if you forget to call `free()` after `malloc()`? a) Compilation error b) Memory leak c) Program crashes immediately d) Nothing happens

Question 12: How do you declare a pointer to an integer? a) `int pointer x;` b) `int *x;` c) `pointer int x;` d) `int &x;`

Question 13: What will this print if run with `./program hello world`?

```
printf("%d", argc);
```

a) 2 b) 3 c) 1 d) 4

Question 14: Which header file do you need for string functions like `strlen()`? a) `<stdio.h>` b) `<string.h>` c) `<stdlib.h>` d) `<strings.h>`

Question 15: In multi-file projects, what are header files (.h) used for? a) Function implementations b) Function prototypes and declarations c) Main function only d) Variable definitions

Question 16: What does `#ifndef` do in header files? a) Includes a file b) Defines a constant c) Prevents multiple inclusions d) Creates a function

---

Section B: Short Answer Questions (28 marks)

Provide clear, concise answers. 4 marks each.

Question 17: Explain the difference between stack and heap memory. Give one advantage of each.

Question 18: What is the purpose of function prototypes in header files? Why can't we just put the full function definitions there?

Question 19: Explain what "memory leak" means and how it occurs in C programs.

Question 20: Why is `fgets()` considered safer than `scanf()` for reading strings?

Question 21: What is the difference between `char str[] = "Hello";` and `char *str = "Hello";`?

Question 22: Explain what happens when you pass an array to a function. How is this different from passing a regular variable?

Question 23: What is the purpose of command line arguments? Give a real-world example of when they would be useful.

Section C: Code Analysis and Completion (20 marks)

Complete or fix the code. 5 marks each.

Question 24: Complete this function that counts words in a string:

```
int count_words(char str[]) {
    int count = 0;
    int in_word = 0;
```

```
    for (int i = 0; str[i] != ______; i++) {
        if (str[i] != ' ' && str[i] != '\t') {
            if (______) {
                count++;
                in_word = ______;
            }
        } else {
            in_word = ______;
        }
    }
    return count;
}
```

Question 25: Fix the memory management issues in this code:

```
char* create_greeting(char *name) {
    char greeting[100];
    sprintf(greeting, "Hello, %s!", name);
    return greeting;  // Problem here!
}
```

Write the corrected version:

Question 26: Complete this command line argument processor:

```
int main(int argc, char *argv[]) {
    if (argc != ______) {
        printf("Usage: %s <number>\n", ______);
        return 1;
    }

    int num = ______(argv[1]);
    printf("Square of %d is %d\n", num, num * num);
    return 0;
}
```

Question 27: Complete this pointer swap function:

```
void swap(int *a, int *b) {
    int temp = ______;
    *a = ______;
    *b = ______;
}
```

Section D: Programming Problems (20 marks)

Question 28: String Processor (10 marks) Write a complete program that:

- Prompts the user for a sentence (use `fgets` for safety)
- Implements a function `void reverse_string(char str[])` that reverses the string in place
- Implements a function `int find_char(char str[], char ch)` that returns the position of a character (or -1 if not found)
- In main: displays the original string, reverses it, shows the reversed string, and finds the position of 'a'

Question 29: Dynamic Array Manager (10 marks) Write a program that:

- Prompts the user for how many integers they want to store
- Dynamically allocates memory for that many integers
- Reads the integers from the user
- Calculates and displays the sum
- Properly frees the allocated memory

Include proper error checking for `malloc()` failure.

Answer Template

Section B - Short Answers:

Q17: _____

Q18: _____

Q19: _____

Q20: _____

Q21: _____

Q22: _____

Q23: _____

Section C - Code Completion:

Q24:

```
int count_words(char str[]) {
    int count = 0;
    int in_word = 0;
```

```
    for (int i = 0; str[i] != ______; i++) {
        if (str[i] != ' ' && str[i] != '\t') {
            if (______) {
                count++;
                in_word = ______;
            }
        } else {
            in_word = ______;
        }
    }
    return count;
}
```

Q25:

```
// Write corrected version here
```

Q26:

```
int main(int argc, char *argv[]) {
    if (argc != ______) {
        printf("Usage: %s <number>\n", ______);
        return 1;
    }

    int num = ______(argv[1]);
    printf("Square of %d is %d\n", num, num * num);
    return 0;
}
```

Q27:

```
void swap(int *a, int *b) {
    int temp = ______;
    *a = ______;
    *b = ______;
}
```

Section D - Programming Problems:

Q28: String Processor

```
// Write your complete program here
```

Q29: Dynamic Array Manager

```
// Write your complete program here
```

Marking Rubric

- **MCQ (32 marks):** 2 marks per correct answer
- **Short Answer (28 marks):** 4 marks each - clear understanding and accurate explanation
- **Code Completion (20 marks):** 5 marks each - correct syntax and logic
- **Programming (20 marks):** 10 marks each
  - Correct function implementation: 6 marks
  - Proper memory/string handling: 3 marks
  - Style and error checking: 1 mark

Key Success Factors:

- Understand pointer dereferencing vs address-of operators
- Remember string null termination
- Check `malloc()` return values
- Free all allocated memory
- Handle command line arguments correctly
- Use safe string functions when possible