Mock Exam 4: Strings, Command Line Arguments, and Pointers

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 '\n' d) Automatically by the compiler

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

Question 3: Which function safely copies one string to another? a) `strcopy()` b) `strcpy()` c) `stringcopy()` d) `copy()`

Question 4: What does `argc` represent in `main(int argc, char *argv[])`? a) The program name b) The number of command line arguments c) The first argument d) The argument array

Question 5: What is `argv[0]` always equal to? a) The first user argument b) The program name c) The number of arguments d) NULL

Question 6: What does the `*` operator do with pointers? a) Gets the address of a variable b) Dereferences a pointer (gets the value it points to) c) Creates a new pointer d) Deletes a pointer

Question 7: What does the `&` operator do? a) Gets the value of a variable b) Gets the address of a variable c) Creates a pointer d) Compares two values

Question 8: What will this code print?

```
char str[] = "World";
printf("Hello %s", str);
```

a) Hello World b) Hello str c) World d) Error

Question 9: How do you declare a pointer to an integer? a) `int pointer x;` b) `int *x;` c) `pointer int x;` d) `int &x;`

Question 10: What happens when you pass a pointer to a function? a) The value is copied b) The address is passed, allowing modification of the original c) Nothing happens d) The pointer is deleted

Question 11: Which function converts a string to an integer? a) `stoi()` b) `atoi()` c) `string_to_int()` d) `convert()`

Question 12: What will this print if run with `./program hello world`?

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

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

Question 13: How do you concatenate two strings? a) `str1 + str2` b) `strcat(str1, str2)` c) `concat(str1, str2)` d) `str1.append(str2)`

Question 14: What does `malloc()` do? a) Allocates memory on the stack b) Allocates memory on the heap c) Frees memory d) Creates a pointer

Question 15: What must you do after using `malloc()`? a) Nothing b) Call `free()` c) Call `delete()` d) Set the pointer to 0

Question 16: What will happen if you forget to call `free()`? a) Program crashes b) Memory leak c) Compilation error d) Nothing happens

---

Section B: Code Analysis and Completion (28 marks)

Question 17: String Function (10 marks) Complete this function that counts the number of vowels in a string:

```
int count_vowels(char str[]) {
    int count = 0;
    for (int i = 0; str[i] != ________; i++) {
        char c = str[i];
        if (c == 'a' || c == 'e' || c == 'i' ||
            c == 'o' || c == 'u' || c == 'A' ||
            c == 'E' || c == 'I' || c == 'O' || c == 'U') {
            ________;
        }
    }
    return ________;
}
```

Question 18: Pointer Swap (10 marks) Complete this function that swaps two integers using pointers:

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

Question 19: Command Line Calculator (8 marks) Complete this program that adds two numbers from command line:

```c
int main(int argc, char *argv[]) {
    if (argc != ________) {
        printf("Usage: %s <num1> <num2>\n", ________);
        return 1;
    }

    int num1 = ________(argv[1]);
    int num2 = ________(argv[2]);
    int sum = num1 + num2;

    printf("%d + %d = %d\n", num1, num2, sum);
    return 0;
}
```

Section C: Programming Problems (40 marks)

Question 20: String Processor (20 marks) Write a complete program that:

1. **Prompts the user for a sentence** (use `fgets` for safety)
2. **Implements these functions:**
   - `int word_count(char str[])` - counts words in the string
   - `void reverse_string(char str[])` - reverses the string in place
   - `int find_char(char str[], char target)` - returns position of character (or -1 if not found)
3. **In main:**
   - Gets input from user
   - Displays original string
   - Shows word count
   - Shows position of the letter 'a' (or -1 if not found)
   - Reverses and displays the reversed string

Example run:

```
Enter a sentence: Hello world
Original: Hello world
Word count: 2
Position of 'a': -1
Reversed: dlrow olleH
```

Question 21: Dynamic Array Manager (20 marks) Write a program that demonstrates dynamic memory allocation:

1. **Prompt the user** for the number of integers they want to store
2. **Allocate memory** dynamically for that many integers

3. **Fill the array** by prompting for each number
4. **Calculate and display:**
    - The sum of all numbers
    - The average (as a double)
    - The maximum value
5. **Free the allocated memory** before the program ends

Include these functions:

- `int* create_array(int size)` - allocates and returns pointer to array
- `int calculate_sum(int *arr, int size)` - returns sum of array elements
- `int find_maximum(int *arr, int size)` - returns maximum value

Example run:

```
How many numbers? 4
Enter number 1: 10
Enter number 2: 20
Enter number 3: 15
Enter number 4: 25

Sum: 70
Average: 17.50
Maximum: 25
```

Remember to:

- Check if `malloc()` returns `NULL`
- Free allocated memory
- Handle edge cases appropriately

Answer Template

Section B Solutions:

Question 17:

```c
// Complete the vowel counting function
int count_vowels(char str[]) {
    int count = 0;
    for (int i = 0; str[i] != ________; i++) {
        char c = str[i];
        if (c == 'a' || c == 'e' || c == 'i' ||
            c == 'o' || c == 'u' || c == 'A' ||
            c == 'E' || c == 'I' || c == 'O' || c == 'U') {
            ________;
        }
    }
```

```
    return ________;
}
```

Question 18:

```
// Complete the swap function
void swap(________ *a, ________ *b) {
    int temp = ________;
    ________ = *b;
    *b = ________;
}
```

Question 19:

```
// Complete the command line calculator
int main(int argc, char *argv[]) {
    if (argc != ________) {
        printf("Usage: %s <num1> <num2>\n", ________);
        return 1;
    }

    int num1 = ________(argv[1]);
    int num2 = ________(argv[2]);
    int sum = num1 + num2;

    printf("%d + %d = %d\n", num1, num2, sum);
    return 0;
}
```

Section C Solutions:

Question 20 - String Processor:

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

Question 21 - Dynamic Array Manager:

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

Marking Rubric

- **MCQ (32 marks):** 2 marks per correct answer
- **Code Completion (28 marks):**
  - Logic correctness: 70%
  - Syntax accuracy: 30%
- **Programming Problems (40 marks):** 20 marks each
  - Correct function implementation: 12 marks
  - Proper memory management: 4 marks
  - Input/output handling: 4 marks

Key Reminders:

- Strings end with '\0'
- Always check `malloc()` return value
- Free dynamically allocated memory
- Use `fgets()` for safe string input
- Command line arguments start with program name at `argv[0]`