Course: Data Structures (CSE CS203A, 114-1)
Quiz I: Introduction to C Programming and Data Structures (with Answer)
September 30, 2025, 16:30~17:00

Student ID: Student Name:

Q1: (20 pts; 5 pts for each) **Complete the C Code**

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    ____①____ *array;
    int n = 10;

    // Allocate memory for n integers
    array = (int *) malloc(n * ____②____ );

    // Initialize array with values 1, 2, 3, ..., 10
    for(int i = 0; i < n; i++) {
        array[i] = i + 1;
    }

    // Print the original array
    printf("Original array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");

    // Double the array size
    n = n * 2;
    array = (int *) ____③____ (array, n * sizeof(int));

    // Initialize new elements (second half)
    for (int i = n/2; i < n; i++) {
        array[i] = i + 1;
    }

    // Print the resized array
    printf("Resized array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");

    // Clean up memory
    _____④_____
    array = NULL;

    return 0;
}
```

A1:

① int

② sizeof(int)

③ realloc

④ free(array)

Q2: (20 pts) **Memory Management Code Review**

You are conducting a code review for a junior developer who submitted the following C code for a production system that will handle user data processing. The code dynamically allocates memory for an double array, processes the data, and then expands the array size as needed.

```
double *array;
int n = 10;

array = (double *) malloc(n * sizeof(double));

// ... processing code ...

n = n * 2;
array = (double *) realloc(array, n * sizeof(double));

// ... more processing ...

free(array);
```

As a senior developer responsible for code quality and system reliability, you notice several critical memory management issues that could lead to:

- Memory leaks
- Segmentation faults
- System crashes in production

- Data corruption
- Undefined behavior

Task: Identify the specific memory management issues and provide solutions to ensure safe memory management.

A2:

- **Missing malloc() error checking**: If malloc() fails and returns NULL, the program will crash when trying to access array elements.
- **Unsafe realloc() usage**: Direct assignment to the original pointer can cause memory leaks if realloc() fails. When realloc() returns NULL, the original memory block is lost.
- **No error handling strategy**: The program continues execution even if memory allocation fails.

```
double *array;
int n = 10;

// Safe malloc with error checking
array = (double *) malloc(n * sizeof(double));

if (array == NULL) {
    fprintf(stderr, "Error: Failed to allocate memory for %d doubles\n", n);
    return 1;
}

// ... processing code ...

// Safe realloc with temporary pointer
n = n * 2;
double *temp = (double *) realloc(array, n * sizeof(double));

if (temp == NULL) {
    fprintf(stderr, "Error: Failed to reallocate memory for %d doubles\n", n);
    free(array); // Clean up original memory
    return 1;
}
array = temp; // Safe to update pointer only after success

// ... more processing ...

free(array);
array = NULL; // Prevent accidental reuse
```

Q3: (40 pts) **Time Complexity Analysis**

Fill in the blanks with the appropriate Big O notation: O(1), O(log n), O(n), O(n log n), $O(n^2)$, $O(n^3)$, O(n!).

Q3-1: (5pts) If binary search is O(log n) and we perform it n times, the overall time complexity is _______.

```
for(int i = 0; i < n; i++) {
        // Binary search operation on sorted array
        binarySearch(sortedArray, target, n);
}
```

Q3-2: (5 pts)

Accessing an element in an array by index (e.g., array[5]) has a time complexity of _______.

Q3-3: (15 pts; 5 pts for each)

Finding the maximum value in an unsorted array by checking every element has a time complexity of _______.

Traversing through all elements in an array of size n has a time complexity of _______.

Do these two operations have the same time complexity? _______ (Yes/No).

Q3-4: (5 pts)

Bubble sort algorithm for sorting an array of n elements has a time complexity of _______.

Q3-5: (10 pts)

Order the following Big O notations from fastest (most efficient) to slowest (least efficient):

Given: $O(n!)$, $O(1)$, $O(n^2)$, $O(\log n)$, $O(n \log n)$, $O(n)$, $O(n^3)$

A3-1: O(n log n)

A3-2: O(1)

A3-3: O(n), O(n), Yes

A3-4: $O(n^2)$

A3-5: $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(n!)$

Q4: (20 pts) **Explain the difficulties in learning data structures.**

Task: Discuss the main challenges students face when learning data structures and suggest approaches to overcome these difficulties.

A4:

Give us your feedback. It’s valuable for me and for the improvement on this course.