

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

Student ID: 1133346

Student Name: 趙子傑

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

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

int main() {
    ____ ① int ____ *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]);
    }
}
```

90 f/10

```

}
printf("\n");

// Clean up memory
_____④_____
array = NULL;

return 0;
}

```

A1:

- ① int
- ② sizeof(int)
- ③ realloc
- ④ free(array);



f2v

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 integer 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: 最好應該加上 `array = NULL;` 這避免之後 dangling fault
 要注意空間是不是足夠 才能 `realloc`, 補上 `if (array == NULL) { cerr("wrong"); }`
 沒有 `#include` 沒有 `int main()` 同時也要注意裡面空間有沒有被其它佔用掉
 undefined behavior / segmentation faults memory leaks
`#include <stdio.h>`
`#include <stdlib.h>` → `realloc` 等等才能做
`int main()`

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 $O(n \log n)$.

```
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 $O(1)$.

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 $O(n)$.

Traversing through all elements in an array of size n has a time complexity of $O(n)$.

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

Q3-4: (5 pts)

Bubble sort algorithm for sorting an array of n elements has a time complexity of $O(n^2)$.

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 ✓

+40

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

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

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: 希望可以把結構寫得更清楚 (在ppt上) 跟更有構的將重點列出來再詳談 (ppt)

+20