

Student ID: 1131428

Student Name: 徐立伟

92/100

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]);
    }
}
```

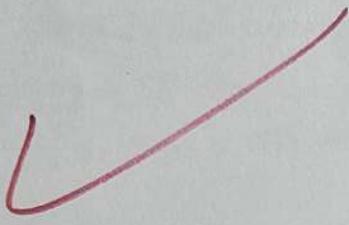
```
    }
    printf("\n");

    // Clean up memory
    —④—
    array = NULL;

    return 0;
}
```

A1:

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



+20.

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:
1. add `if(array==null){ printf("no enough space"); return 0; }` after malloc & realloc
 2. add `array=null; return 0;` after free(array)
 3. add `#include <stdio.h>` at front of code
~~`#include <stdlib.h>`~~
 4. add `int main(){...}` after `#include`, ^{other} code must inside the {}.

Q3: (40 pts) Time Complexity Analysis

+12

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);
}
```

$(\log n)^n$

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)

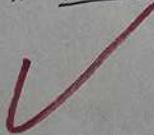
Bubble sort algorithm for sorting an array of n elements has a time complexity of $O(n^2)$.

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: $O(n \log n)$



140

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:

困難：① 理解結構的規則

ex: 紅黑樹有好多細項……

② 從理論轉換成 code

ex: link list 的圖可以理解，變成 code 時…… 為什麼 struct? → ? A → B 代表的意思??

③ 實作時出 bug 不知道從那裡改

ex: 這一行一直冒紅但 why? 我那裡錯了!?

解決法: ① → 尋試和詳細的說明，可以幫助理解
來自老師&同學的

② 實作可以加深印象，但可能需要一段時間

③ ~~uncopiler & chatgpt~~ 問同學及老師，多錯幾次就不會錯了但過程很痛苦。

+ 20