Princess Sumaya جامعة الأميرة سميّة للتكنولوجيا University for Technology

The King Hussein School for Computing Sciences
Department of Computer Science
11103 - Structured Programming - Fall 2023

Final Exam

Full Name: Student ID:

Question	Points	Score
1	5	
2	9	
3	4	
4	6	
5	6	
6	10	
Total	40	

Circle your section:

- ○ Dr. Rawan Ghnemat (section 1)
- ○ Dr. Rawan Ghnemat (section 2)
- ○ Dr. Mohammad Abu Snober (section 3)
- ○ Dr. Rawan Ghnemat (section 4)
- ○ Dr. Mohammad Abu Snober (section 5)

Monday 5/2/2024

Question 1. Basic Code Reading and Debugging (5 marks)

Find the output of each row independently. Use **ERROR** if the code leads to a compilation or a runtime error. Assume that x, y and a are **global variables** defined as: `int x = 1, y = 2, a[3] = {9, 1, 5};`

	CODE	OUTPUT

1	`x = y;`
	`y = x;`
	`printf("%d %d", x, y);`	

2	`int b[3] = {1};` `printf("%d", b[2]);`	
3	`if (x < 1) printf("A");` `else (x >= 1) printf("B");`	
4	`x = 1.5;` `printf("%f", x);`	

5	`if (x = 2)`
	`  printf("A ");`
	`printf("B ");`	

6	`while (x--) printf("%d ", x);`	

7	`for (int i = 0; i < 3;  i++) {`
	`  if (i == 0) continue;`
	`  if (i == 2) break;`
	`  printf("A");`
	`}`	

8	`int* ptr = &a[0];` `printf("%d", *ptr+1);`	

9	`int* ptr1 = &x;`
	`int* ptr2 = &y;`
	`ptr1  = ptr2;`
	`*ptr1 = *ptr2;`
	`printf("%d %d", x, y);`	

10	`if (x == 2)`
	`if (y == 1)`
	`printf("A");`
	`else printf("B");`
	`printf("C");`	

Question 2. Basic Code Writing (9 marks)

Implement each of the following functions.

A. Returns `1` if at least one argument is positive and at least one argument is negative, and `0` otherwise.

```
int diff_sign(int a, int b, int c) {




}
```

B. Prints the pattern: `1 100 2 99 3 98 4 97 5 96 ... 99 2 100 1`

```
void print_pattern() {




}
```

C. Prints the pattern: `1-  2--  3---  4----  5----- ...` (assuming `n > 0`)

```
int n_dashes(int n) {          // the last term printed is n followed by n dashes




}
```

Question 3. Code Comprehension (4 Marks)

```
void fun(int a[], int N) {
    for (int i = 0; i < N; i++) {
        printf("A B ");

        int sum = 0;
        for (int j = 0; j < N; j++) {
            sum += a[j];
            printf("B ");
        }

        a[i] = sum;
    }
}
```

Answer the questions below assuming array `a[]` contains the value `2` in all the array cells.

A. `[2 marks]` Assume that `N=1`, and that `a[]` is of size `1`.

- How many times will "**A**" be printed out? ____
- How many times will "**B**" be printed out? ____
- What will be stored at `a[0]` after the function finishes execution? ____

B. `[2 marks]` Assume that `N=100`, and that `a[]` is of size `100`.

- How many times will "**A**" be printed out? ____
- How many times will "**B**" be printed out? ____
- What will be stored at `a[0]` after the function finishes execution? ____

Question 4. Recursion (6 marks)

1. [2 marks] Complete the following *recursive* function, which reverses the subarray `a[i → j]`.

Example. If the elements in `a[]` are: `{0, 1, 2, 3, 4, 5, 6, 7}`
Calling the function using `i = 2` and `j = 6` makes the array:
`{0, 1, 6, 5, 4, 3, 2, 7}`

You can assume that `i` and `j` are valid indices and that a is defined globally.

```
void reverse(int i, int j) {
   if (i >= j) return;

   _____________________________________;

   _____________________________________;
}
```

2. [2 marks] Complete the following *recursive* function, which returns `n % m`.

Notes.
- You are not allowed to use the `%` or `/` operators (**Hint:** use subtraction!).
- You can assume that `n` and `m` are greater than `0`.

```
int remainder(int n, int m) {
   if (n < m) return _____________________________________;

   else       return _____________________________________;
}
```

3. Answer the questions on the right about the following function.

```
void fun(int n) {
   if (n <= 0)
      return;

    printf("%d ", n);
   fun(n-1);
   fun(n-2);
}
```

1. [0.5 mark] What is the output of calling `fun(1)` ?

2. [0.5 mark] What is the output of calling `fun(2)` ?

3. [1 mark] What is the output of calling `fun(4)` ?

Question 5. Strings and Pointers (6 marks)

Implement function `void remove_duplicates(char* str, char* result)`, which copies `str` to `result` after removing duplicate characters that are next to each other.

Examples.

str	**result**
aaabbbaaa	aba
ababab	ababab
helloooo, there!!!	helo, there!

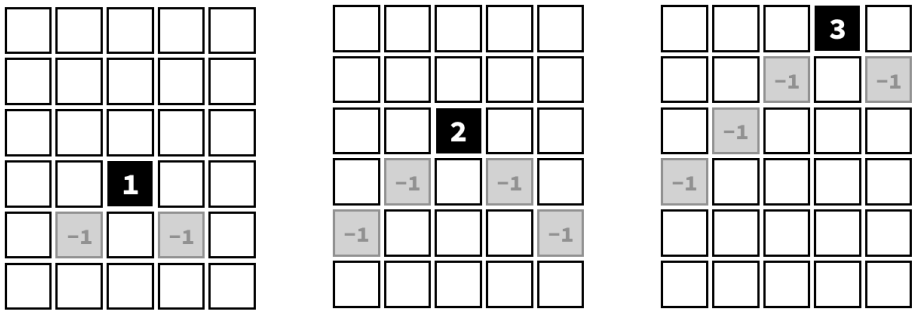
Notes.

- You can assume that no string will be longer than `100` characters.
- You are not allowed to use the array `[]` notation. You must use pointer arithmetic only.
- You are not allowed to use the `string.h` library.

Question 6. 2D Arrays (10 marks)

1. [5 marks] Implement function **void** blow_up2(**int** a[N][M], **int** r, **int** c), which receives a 2D array of size NxM (where N and M are defined globally) and an index (i, j) that needs to be blown up!

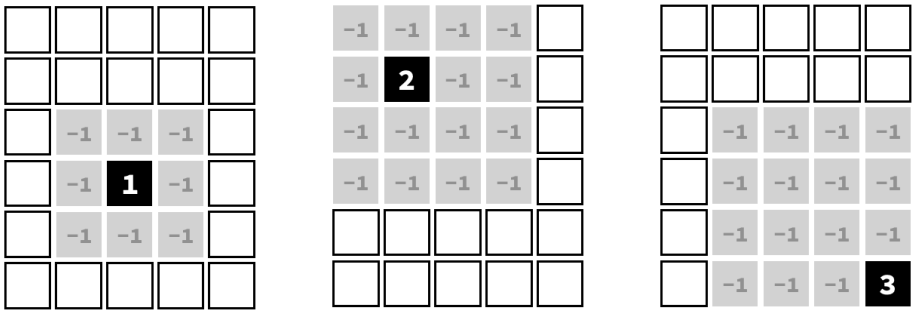
When a cell a[i][j] blows up, it destroys the cells to its bottom-left and bottom-right. The value stored at cell a[i][j] is the length of the diagonals that will be destroyed, as shown in the examples below.



Implement function blow_up by storing the value -1 in the cells that must be destroyed.

2. [5 marks] Implement function **void** blow_up2(**int** a[N][M], **int** r, **int** c), which receives a 2D array of size NxM (where N and M are defined globally) and an index (i, j) that needs to be blown up!

When a cell a[i][j] blows up, it destroys the cells around it. The value stored at cell a[i][j] describes the radius around a[i][j] that will be destroyed as shown in the examples below.



Implement function blow_up2 by storing the value -1 in the cells that must be destroyed.