Princess Sumaya University for Technology
جامعة الأميرة سميّة للتكنولوجيا

The King Hussein School for Computing Sciences
Department of Computer Science

Structured Programming - Spring 2022

Second Exam

Full Name: `Reference Solution` **Student ID:**

Circle your section:

○ Dr. Mu'awya Al-Dala'ien	(section 1)
○ Dr. Rawan Ghnemat	(section 2)
○ Dr. Abdullah Aref	(section 3)
○ Dr. Mu'awya Al-Dala'ien	(section 4)
○ Dr. Rawan Ghnemat	(section 5)
○ Dr. Sawsan Alshatnawi	(section 6)
○ Dr. Mohammad Al Nabhan	(section 7)
○ Dr. Sawsan Alshatnawi	(section 8)
○ Dr. Mohammad Abu Snober	(section 9)
○ Dr. Mohammad Abu Snober	(section 10)
○ Dr. Khaled Mansour	(section 11)
○ Dr. Abedalrhman Alkhateeb	(section 13)
○ Dr. Khaled Mansour	(section 14)
○ Dr. Rafat Hammad	(section 15)

Question	Points	Score
1	4	
2	3	
3	4+1	
4	7	
5	7	
Total	25+1	

Question 1 (4 points)

Fill the **Output** column in the table below with the output of the code provided in the **Code** column. If the code does not compile, write “**compilation error**” instead of the output.

	Code	Output

1.	`int a[2][2] = {{1, 2}, {3, 4}};`
	`printf("%d", a[0][1]);`
		2

2.	`int a[3][3] = {{1, 2}, {3, 4}};`
	`printf("%d", a[2][2]);`
		0

3.	`int a[][] = {{1, 2}, {3, 4}};`
	`printf("%d", a[1][1]);`
		Compilation Error

4.	`int x = 2;`
	`do printf("%d ", x--);`
	`while (x >= 2);`
		2

5.	`for (int i = 0; i < 3; i++)`
	`    if (i == 1) continue;`
	`    else         printf("%d ", i);`
		0 2

6.	`for (int i = 0; i < 3; i++)`
	`    if (i == 1) break;`
	`    else         printf("%d ", i);`
		0

7.	`for (int i = 0; i < 2; i++)`
	`    printf("%d ", i);`
	`for (int j = 0; j < 2; j++)`
	`    printf("%d ", j);`
		0 1 0 1

8.	`void f(int x) {`
	`    if (x == 3) break;`
	`    else         printf("Hello");`
	`}`
	
	`int main() {`
	`    f(3);`
	`    return 0;`
	`}`
		Compilation Error

Question 2 (3 points)

Convert the following function to a recursive function:

```
void boom(int n) {
    while (n > 0)
        printf("%d ", n--);
    printf("Boooom!");
}
```

```
void boom(int n) {
    if (n <= 0) {
        printf("Boooom!");
        return;
    }
    printf("%d ", n);
    boom(n-1);
}
```

Question 3 (4+1 points)

PART 1.

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n - 1; j++) {
        if      (a[i][j] == a[i][j+1]) printf("A ");
        else if (a[i][j] == a[i][j-1]) printf("B ");
    }
}
```

A. Provide an example of an array `a[][]` of size `[n=3]x[n=3]` that will cause the above code to print `A A A A A A.`

```
1 1 1
1 1 1     any 3x3 array whose elements are all
1 1 1     the same is a correct answer.
```

B. Provide an example of an array `a[][]` of size `[n=3]x[n=3]` that will cause the above code to print `A A A A A B.`

```
1 1 1
1 1 1     any 3x3 array whose elements are all
1 1 0     the same except the last is a correct answer.
```

C. [+1 point] Provide an example of an array `a[][]` of size `[n=3]x[n=3]` that might cause the above code to crash.

```
0 1 1     any 3x3 array with an element at the
1 1 1     first column != the element to its
1 1 1     right causes the code to access a
          negative index which might crash.
```

Note. This part is a **bonus** question. Do **not** spend time on it until you are done with the other required questions.

PART 2.

```
void f1(int a[], int n) {

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n - 1; j++) {
            int temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
}
```

D. What are the contents of array `a[]` after calling function **`f1`** if **`n = 2`** and **`a[] = {2, 1}`** ?

`{2, 1}`

E. What are the contents of array `a[]` after calling function **`f1`** if **`n = 100`** and **`a[] = {100, 99, 98, 97, …, 3, 2, 1}`** ?

`{100, 99, 98, 97, …, 3, 2, 1}`

Question 4 (7 points)

A. [4 points] Implement a function named **sudoku**, which receives as an argument a `2D` array of integers of size `9x9`. The function returns **`1`** if every column sums to **`45`** and every row sums to **`45`**. The function returns **`0`** otherwise.

```
int sudoku(int a[][9]) {
    for (int i = 0; i < 9; i++) {
        int sum = 0;
        for (int j = 0; j < 9; j++)
            sum += a[i][j];
        if (sum != 45)
            return 0;
    }

    for (int j = 0; j < 9; j++) {
        int sum = 0;
        for (int i = 0; i < 9; i++)
            sum += a[i][j];
        if (sum != 45)
            return 0;
    }
    return 1;
}
```

B. [3 point] Write a program that creates a 2D array of size 9x9, fills it with random integers between 1 and 9 (inclusive) and then uses function **sudoku** to check if every row and every column in the array sums to 45. If this is true, your program must print "what a surprise!".

```
int main() {
    int a[9][9];
    for (int i = 0; i < 9; i++)
        for (int j = 0; j < 9; j++)
            a[i][j] = 1 + rand() % 9;

    if (sudoku(a))
        print("what a surprise!");

    return 0;
}
```

Question 5 (7 points)

In Number Theory, a Taxicab Number is a number that can be expressed as a sum of cubes in *more than one way*. For example, 1729, 4104 and 13832 are taxicab numbers, because:

$$\begin{array}{rcl} 1729 = 1^3 + 12^3 & \text{and also} & 1729 = 10^3 + 9^3 \\ 4104 = 2^3 + 16^3 & \text{and also} & 4104 = 9^3 + 15^3 \\ 13832 = 20^3 + 18^3 & \text{and also} & 13832 = 24^3 + 2^3 \end{array}$$

A. [5 points] Implement a function named **taxicab** that receives an integer and prints "taxicab" if the integer is a taxicab number and "not taxicab" otherwise.

```
void taxicab(int n) {
    int count = 0;
    for (int i = 1; i < n; i++)
        for (int j = i; j < n; j++)
            if (i*i*i + j*j*j == n)
                count++;
    if (count > 1)
        printf("taxicab");
    else
        printf("not taxicab");
}
```

B. [2 points] Reimplement function **taxicab** such that it prints all the taxicab numbers that are less than the received integer.

```
void taxicab(int k) {
    for (int n = 1; n < k; n++) {
        int count = 0;
        for (int i = 1; i < n; i++)
            for (int j = i; j < n; j++)
                if (i*i*i + j*j*j == n)
                   count++;
       if (count > 1)
           printf("%d\n", n);
   }
}
```

Note 1. No double-jeopardy. Grade part B only based on putting the code from part A into a loop correctly. Do not deduct in part B for errors already deducted for in part A.

Note 2. Students might optimize the code by looping until n/2, n/3, or sqrt(n), etc. instead of to n. This is all correct.

Note 3. The innermost loop must start from j = i not from j = 0. Starting from j = 0 means that the same pair will be counted twice (e.g. 10 9 and then 9 10). Apply a minor deduction for this error.