Princess Sumaya جامعة الأميرة سميّة للتكنولوجيا University for Technology

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

First Exam

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

Question	Points	Score
1	5	
2	4	
3	PART 1: 4	
	PART 2: 4	
	PART 3: 4	
	PART 4: 4	
Total	25	

Circle your section:

- ○ Dr. Ammar Alrashdan (section 1)
- ○ Dr. Osama Alhaj Hasan (section 2)
- ○ Dr. Rawan Ghnemat (section 3)
- ○ Dr. Ammar Alrashdan (section 4)
- ○ Dr. Rawan Ghnemat (section 5)
- ○ Dr. Mohammad Al Nabhan (section 6)
- ○ Dr. Mohammad Al Nabhan (section 7)
- ○ Manaf Gharaibeh (section 8)
- ○ Dr. Mohammad Abu Snober (section 9)
- ○ Dr. Mohammad Abu Snober (section 10)
- ○ Mr. Yousef Yaseen (section 11)
- ○ Mr. Alaa Altarazi (section 12)
- ○ Mr. Alaa Altarazi (section 13)
- ○ Mr. Alaa Altarazi (section 14)

Saturday 1/4/2023

Question 1 (5 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.

Assume `x` and `a[]` are defined as follows: `int x = 5;` `int a[10] = {1, 2, 3, 4};`
Assume also that each row is independent (does not depend on the rows before).

	Code	Output
1.	`printf("%d", 3 + 1 / 2 * 2 - 2);`	1
2.	`x == 1 ? printf("1") : printf("0");`	0
3.	`printf("%d", a[5]);`	0
4.	`printf("%0.3f", x + 0.5);`	5.500
5.	`printf("%d", 'b' - 'a');`	1
6.	`if (rand() % x == 5) printf("5");` `else printf("??");`	??
7.	`while (x > 1) printf("%d", --x);`	4321

8.	`#define Y = 5;`
	`int main() { printf("%d", Y+1);`
	`return 0; }`
		Compilation Error

9.	`if (x = 1) printf("equal 1");` `else printf("not equal 1");`	equal 1

10.	`int y = 0;`
	`void f(int y) {`
	`printf("%d ", y);`
	`}`
	
	`int main() {`
	`f(6);`
	`printf("%d ", y);`
	`return 0;`
	`}`
		6 0

Question 2 (4 points)

What is the output of the following piece of code in each of the cases given below? Assume that `N` was defined using `#define`.

```
// define and initialize an array
// a[] of size N

int i = 0;
while (i < N && a[i] % 2 != 0)
      i++;

int j = N-1;
while (j >= 0 && a[j] % 2 != 0)
      j--;

if (j <= i || a[i] != a[j])
      printf("-1");
else
      printf("%d", a[i]);
```

A. If `N = 1` and `a = {2}`

-1

B. If `N = 2` and `a = {4, 4}`

4

C. If `N = 1000` and `a = {0, 1, 2, ..., 999}`

-1

D. If `N = 1000` and `a = {997, 995, 993, ..., 5, 3, 1, 0, 1, 3, 5, ..., 999}`

-1

Question 3 (16 points)

PART 1. Write a **void** function named **checkFermat** that receives four integers: a, b, c, and n.
The function prints:

- `"Fermat is wrong"` if $a^n + b^n = c^n$
- `"Fermat might be correct"` otherwise.

You are **not** allowed to use any function from `math.h`.

```
void check_fermat(int a, int b, int c, int n) {
    long an = 1, bn = 1, cn = 1;
    for (int i = 0; i < n; i++) {
        an *= a;
        bn *= b;
        cn *= c;
    }

    if (an + bn == cn) printf("Fermat is wrong!");
    else               printf("Fermat might be correct!");
}
```

PART 2. Implement a **void** function named **hangman**, which receives a `word` as a character array and a letter `chr`. The function prints the word hiding the letters that are not equal to `chr` (using `'_'`).

Examples.

		After calling `hangman(word, chr)`
• `word = "international",`	`chr = "n"`	`word = "_n___n____n__"`
• `word = "woow",`	`chr = "o"`	`word = "_oo_"`
• `word = "hello",`	`chr = "z"`	`word = "_____"`

```
void hangman(char s[], char chr) {
    for (int i = 0; s[i] != '\0'; i++) {
        if (s[i] != chr) printf("_");
        else             printf("%c", s[i]);
    }
}
```

PART 3. Implement a function named **is_triplets**, which receives an array of integers and its size. The function returns `1` if the array is made of triplets of equal numbers (see the examples) and `0` otherwise.

Examples.

- [5, 5, 5, 2, 2, 2, 1, 1, 1] return 1
- [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] return 1
- [5, 5, 5, 2, 2, 2, 1, 1] return 0
- [1, 2, 3, 1, 2, 3, 1, 2, 3] return 0

```
int is_triplet(int a[], int N) {
    if (N % 3 != 0)   // 1 point out of 4
        return 0;

    for (int i = 0; i < N; i += 3) {
        if (a[i] != a[i+1] || a[i] != a[i+2])
            return 0;
    }
    return 1;
}
```

PART 4. Implement a function named **majority_odd**, which receives three integers and returns `1` if the majority of these integers are odd and `0` otherwise. (3.5 points)

Examples: `1, 2, 3:` Return `1` because two of the numbers are odd.
`2, 4, 5:` Return `0` because only one of the numbers is odd.

Note. You are not allowed to define any variable in your implementation. If you do, you will receive at most 2/3.5 points.

```
int majority_odd(int a, int b, int c) {
     if (a%2 != 0 && b%2 != 0 && c%2 != 0) return 1;
     if (a%2 != 0 && b%2 != 0)             return 1;
     if (a%2 != 0 && c%2 != 0)             return 1;
     if (c%2 != 0 && b%2 != 0)             return 1;
     return 0;
}
```

PART 4.5 Implement function **majority_odd** using only one line, containing one statement. (0.5 points)

```
int majority_odd(int a, int b, int c) {
     return (a%2 != 0) + (a%2 != 0) + (a%2 != 0) > 1;
}
```

Another Solution:

```
return      (a%2 != 0 && b%2 != 0 && c%2 != 0) ||
            (a%2 != 0 && b%2 != 0) ||
            (a%2 != 0 && c%2 != 0) ||
            (c%2 != 0 && b%2 != 0);
```