Princess Sumaya جامعة الأميرة سميّة للتكنولوجيا University for Technology

The King Hussein School for Computing Sciences
Department of Computer Science
**11103 - Structured Programming - Spring 2024**

# Midterm Exam

Full Name: Student ID:

| Question | Points | Score |
|---|---|---|
| 1 | 8 | A: |
| | | B: |
| | | C: |
| 2 | 5 | |
| 3 | 10 | |
| 4 | 7 | |
| **Total** | 30 | |

**Circle your section:**

- ○ Dr. Rawan Ghnemat (SuTuTh 9–10)
- ○ Dr. Rawan Ghnemat (SuTuTh 12–13)
- ○ Dr. Rawan Ghnemat (MoWe 8–9½)
- ○ Dr. Samer Sawalha (SuTuTh 8–9)
- ○ Dr. Samer Sawalha (SuTuTh 10–11)
- ○ Dr. Ahmad Klaib (SuTuTh 11–12)
- ○ Dr. Ahmad Klaib (SuTuTh 14–15)
- ○ Dr. Ahmad Klaib (MoWe 12½–14)
- ○ Mr. Alaa Altarazi (SuTuTh 13–14)
- ○ Mr. Alaa Altarazi (MoWe 9½–11)
- ○ Mr. Alaa Altarazi (MoWe 11–12½)
- ○ Dr. Osama Alhaj Hasan (MoWe 11–12½)
- ○ Dr. Mohammad Abu Snober (SuTuTh 11–12)
- ○ Dr. Abdullah Aref (MoWe 14–15½)
- ○ Dr. Amer Al-Badarneh (SuTuTh 12–13)

Monday 15/4/2024

## Question 1 (8 points)

**A.** [4 points] Write a program that repeatedly reads two integers and prints their sum. The program must stop when the user enters two zeroes.

**Example:**

```
Enter two numbers: 1 2
The sum is: 3
Enter two numbers: 0 5
The sum is: 5
Enter two numbers: 0 0
Good bye!
```

EXAMPLE SOLUTION 1:

```
int x;
int y;

printf("Enter two numbers: ");
scanf("%d%d", &x, &y);

while (x != 0 || y != 0) {
    printf("The sum is: %d\n", x+y);
    printf("Enter two numbers: ");
    scanf("%d%d", &x, &y);
}

printf("Good bye!");
```

EXAMPLE SOLUTION 2:

```
int x, y;

while (true) {
    printf("Enter two numbers: ");
    scanf("%d%d", &x, &y);

    if (x == 0 && y == 0)
       break;

    printf("The sum is: %d\n", x+y);
}

printf("Good bye!");
```

**B.** [3 points] Implement a function named **`fillEven`**. The function receives an array of integers and its size. The function must fill the array with *random even* integers in the range [0, 100].

EXAMPLE SOLUTION 1:

```
void fillEven(int a[], int SIZE) {
    for (int i = 0; i < SIZE; i++) {
        do a[i] = rand() % 101;
        while (a[i] % 2 != 0);
    }
}
```

EXAMPLE SOLUTION 2:

```
void fillEven(int a[], int SIZE) {
    int i = 0
     while (i < SIZE) {
          a[i] = rand() % 101;
          if (a[i] % 2 == 0) i++;
     }
}
```

**C.** [1 point] Write a piece of code that defines an array of size 50 and calls function **`fillEven`** to fill the array with random even numbers in the range [0, 100].

```
int a[50];
fillEven(a, 50);
```

## Question 2 (5 points)

Show the output of the code on the right in each of the cases shown below.

**1.** If **N** = 1 and **a**[] = "a"

**2.** If **N** = 2 and **a**[] = "ab"

**3.** If **N** = 1000 and **a**[] = "ffffffff ..."

**4.** If **N** = 1000 and **a**[] = "BeBositiveBeBositiveBeBositiveBeBositive ..."

```
for (int i = 0; i < N; i++)
   for (int j = i + 1; j < N; j++)
       if (a[i] == a[j])
           a[j] = '\0';

int count = 0;

for (int i = 0; i < N; i++)
    if (a[i] != '\0')
        count++;

printf("%d", count);
```

## Question 3 (10 points)

Implement a function named **removeChar**, which receives a string (null-terminated array of characters), and a character `chr`. The function must remove the first occurrence of the character `chr` from the string and return the index of the removed character. If `chr` is not in the string, the function returns `-1`.

**Examples:**

| str before | chr | str after | return value |
|---|---|---|---|
| "programming" | 'r' | "pogramming" | 1 |
| "programming" | 'z' | "programming" | -1 |
| "a" | 'a' | "" | 0 |

**Note**. You are not allowed to use the library `<string.h>`

Provide your answer in the following page.

```
int removeChar(char str[], char chr) {

  int i = 0;
  while (str[i] != '\0') {

    if (str[i] == chr) {
      int j = i;
      while (str[j] != '\0') {
        str[j] = str[j + 1];
        j++;
      }
      return i;
    }

    i++;
  }

  return -1;
}
```

## Question 4 (7 points)

**A.** [2 points] Implement a function named **numLength**, which receives a positive integer x and returns the number of digits in x. For example if `x = 549`, the function returns `3`, and if `x = 9`, the function returns `1`.

**B.** [5 points] Implement a function named **generate**, which receives two positive integers: `seed` and `N`. The function prints a sequence of `N` numbers generated from the `seed` as follows:

- Multiply `seed` by `127`.
- Remove the first and last digits of the result.
- Print the resulting number and use it as the new `seed` for the next iteration.

**Examples:**

**seed = 7, N = 5:**

```
7 x 127 = 889  ⟶ 8
8 x 127 = 1016 ⟶ 1
1 x 127 = 127  ⟶ 2
2 x 127 = 254  ⟶ 5
5 x 127 = 635  ⟶ 3
```

**seed = 713, N = 4:**

```
713 x 127 = 90551 ⟶ 55
55  x 127 = 6985  ⟶ 98
98  x 127 = 12446 ⟶ 244
244 x 127 = 30988 ⟶ 98
```

**seed = 4, N = 6:**

```
4 x 127 = 508 ⟶ 0
0 x 127 = 0   ⟶ 0
0 x 127 = 0   ⟶ 0
0 x 127 = 0   ⟶ 0
0 x 127 = 0   ⟶ 0
0 x 127 = 0   ⟶ 0
```

```
void generate(int seed, int N) {
  for (int i = 0; i < N; i++) {
    seed = seed * 127;
    seed /= 10;                    // remove the first digit

    int x = seed;                  // count the number of digits
    int c = 0;
    while (x > 10) {
      x /= 10;
      c++;
    }
    seed = seed % (int) pow(10, c); // remove the last digit
    printf("%d ", seed);
  }
}
```