Princess Sumaya University for Technology
جامعة الأميرة سميّة للتكنولوجيا

King Hussein School for Computing Sciences
Department of Computer Science
11103 - **Structured Programming** - Fall 2022

Final Exam

Full Name: **Student ID:**

Question	Points	Score
1	15	
2	25	
3	10	
4	13	
5	17	
6	20	
Total	100	

Circle your section:

○ Dr. Rawan Ghnemat	(section 1)
○ Dr. Rawan Ghnemat	(section 2)
○ Dr. Mohammad Abu Snober	(section 3)
○ Ms. Rahmah Ibrahim	(section 4)
○ Mr. Alaa Altarazi	(section 5)

26/1/2023

Question 1. Mini Tracing (15 points)

Find the output of each row independently. If the code does not compile, write **"compilation error".**
Assume that x, y and a are **global variables** defined as: `int x = 1, y = 2, a[3] = {9, 1, 5};`

CODE	OUTPUT
`printf("%d", x++);`	1
`printf("%d", a[0] + a[1] / a[2]);`	9
`printf("%d", (rand()%a[0]) / a[0]);`	0

`if (!(x == y || x == a[0]))  printf("%d", x);`
`else                            printf("%d", y);`
	1

`while (x != y) {`
`    printf("%d", x);`
`    x++;`
`} else {`
`    printf("%d", y);`
`}`
	compilation error

`int *ptr1 = &x, *ptr2 = &y;`
`int *temp = ptr1;`
`ptr1 = ptr2;`
`ptr2 = temp;`
`printf("%d %d", x, y);`
	1 2

`for (int j = 0; j < 3; j++) {`
`    printf("%d ", a[j]);`
`    continue;`
`}`
	9 1 5

`for (int i = 0; i < 3; i++)`
`    for (int j = 0; j < 3; j++) {`
`        printf("%d ", a[j]);`
`        break;`
`    }`
	9 9 9

`void f(int* ptr) { printf("%d", *ptr); }`
`int main()        { f(a); return 0; }`
	9

`if (1 = x) printf("TRUE");`
`printf("FALSE");`
	compilation error

Question 2. Mini Writing (25 points)

Implement each of the following functions.

(A)

```
// Returns 1 if at least two of the
// arguments are non-negative
// Returns 0 otherwise.
int majority(int a, int b, int c) {

    return (a >= 0 && b >= 0) ||
           (a >= 0 && c >= 0) ||
           (b >= 0 && c >= 0);

}
```

(B)

```
// Prints the main diagonal (القطر)
// of the 2D array (top-left to
// bottom-right)
void print_diag(int a[20][20]) {

    for (int i = 0; i < 20; i++) {
       printf("%d ", a[i][i]);
    }

}
```

(C)

```
// Shift every element to the left.
// Discard the first element.
// Store 0 in the last element.
// [1, 2, 3, 4] becomes [2, 3, 4, 0]
void shift_left(int a[], int size) {

    for (int i = 0; i < size-1; i++) {
       a[i] = a[i+1];
    }
    a[size-1] = 0;

}
```

(D)

```
// Returns randomly either 10 or 20.
int rand_10_or_20() {

    if (rand() % 2 == 0)
        return 10;
    else
        return 20;

}
```

(E)

```
// Returns 1 if all the digits in n are even.
// Assume n is non-negative and 0 to be even.
int all_even(int n) {

    while (n > 0) {
        int d = n % 10;
        if (d % 2 != 0)
            return 0;
        n /= 10;
    }
    return 1;

}
```

Question 3. Code Reading (10 points)

What is the output of the function below in each of the given cases? If the function goes into an infinite loop, crashes or does not compile, write **ERROR** instead of the output.

```
void fun(int a[], int size) {
   int i = 0;
   int j = size-1;

   while (i != j) {
       a[i] = a[j];
       a[j] = a[i];

       i++;
       j--;
   }

   for (int i = 0; i < size; i++)
       printf("%d ", a[i]);
}
```

A. If `size = 1` and `a = {1}`

`output = 1`

B. If `size = 3` and `a = {1, 2, 3}`

`output = 3 2 3`

C. If `size = 9999` and `a = {1, 1, 1, ..., 1, 1, 1,  2, 2, 2, ..., 2, 2, 2, 2}`
(`------ 4999 1s ------` over the 1s; `------- 5000 2s --------` over the 2s)
(show only the first 4 values of and the last 4 values of the output)

`output = 2 2 2 2  ...  2 2 2 2`

D. If `size = 9999` and `a = {1, 2, 3, 4, ..., 9996, 9997, 9998, 9999}`
(show only the first 4 values of and the last 4 values of the output)

`output = 9999 9998 9997 9996 ... 9996 9997 9998 9999`

E. If `size = 10000` and `a = {1, 1, 1, 1, 1, ..., 1}`
(show only the first 4 values of and the last 4 values of the output)

ERROR

Question 4. Recursion (13 points)

Implement a *recursive* function **`int find_power2(…)`** that receives an integer $n > 0$ and finds the largest power of 2 $\leq n$.

Examples.

	- $n = 10$	return value = 8
	- $n = 16$	return value = 16
	- $n = 1$	return value = 1 (2^0)
	- $n = 300$	return value = 256

`[10 points]` Provide your implementation in the box below. You are allowed to pick whatever parameters you find suitable.

```
int find_power2(int i, int n) // list the parameters
{
    if (i == n)
        return n;
    if (i  > n)
        return i / 2;
    else
        return find_power2(i * 2, n);
}
```

`[3 points]` Show in the box below how your function is called if $n = 1000$.

```
int result = find_power2(1, 1000);
```

Question 5 (17 points)

Suffix	=	Characters at the end of a string.
Prefix	=	Characters at the beginning of a string.
Weed	=	A suffix and a prefix that are the reverse of each other.

Implement function `void weed_out(const char* str)`, which receives a string and prints it without the weed. Read the following examples carefully to understand more:

Examples.

```
           STRING    OUTPUT
     aaaabcdefaaaa    bcdef
              _ok_    ok
    helloTHEREolleh   THERE
         finalexam    finalexam      // no weed
        hellolleh                    // no output
```

You are not allowed to use array `[]` notation in your implementation. You must use pointer arithmetic only. You are also not allowed to use the `string.h` library.

```
void special_trim(const char* str) {
    const char* end = str;
    while (*end != '\0')
        end++;
    end--;

    while (*str != '\0' && str != end && *str == *end) {
        str++;
        end--;
    }

    while (*str != '\0' && str != end) {
        printf("%c", *str);
        str++;
    }

    if (*str != '\0')
        printf("%c\n", *str);
}
```

Question 6 (20 points)

Pac-Man can see only to his left and to his right. How many monsters can he see?

The game map is represented as a 2D array of characters, where the following characters are used:

`'P'`	The cell where **pac-man** currently is.
`'.'`	A cell containing **food**.
`'M'`	A cell with a **monster**.
Any other character	A **blocked** cell (pac-man can't see through it)

Implement function `int visible_monsters(char map[ROWS][COLS])` that receives the game map and returns the number of monsters pacman can see.

Assume that `ROWS` and `COLS` are defined and globally accessible in the function.

Note that you are not given the location of Pac-Man on the map. You need to find him!
You can assume that there is only one **P** on the map.

Examples

```
. + - - - + P + - - - + M + - - +
. |       | . |       | . |     |
. |       | . + - - - + . |     |
. + - - + | M . . . M . M |     |
. . . . | | . + - + . . . + - + |
+ - + . + + . |   | . . . . . | |
|   | . . . . |   + - - + . + + |
|   + - - + . |         | . |   |
|         | M |   + - - + . |   |
+ - - - - + . + - + . . . . + - +
  Pac-man can't see any monster
```

```
. + - - - + . + - - - + M + - - +
. |       | . |       | . |     |
. |       | . + - - - + . |     |
. + - - + | M . P . M . M |     |
. . . . | | . + - + . . . + - + |
+ - + . + + . |   | . . . . . | |
|   | . . . . |   + - - + . + + |
|   + - - + . |         | . |   |
|         | M |   + - - + . |   |
+ - - - - + . + - + . . . . + - +
  Pac-man can see 3 monsters
```

The solution is on the next page.

```
int visible_monsters(char map[ROWS][COLS]) {
     // find packman
     int i, j;
     for (i = 0; i < ROWS; i++) {
          for (j = 0; j < COLS; j++)
               if (map[i][j] == 'P')
                    break;
          if (j < COLS)
               break;
     }

     // count right
     int count = 0;
     for (int k = j + 1; k < COLS; k++)
          if (map[i][k] == 'M')
               count++;
          else if (map[i][k] != '.')
               break;

     // count left
     for (int k = j - 1; k >= 0; k--)
          if (map[i][k] == 'M')
               count++;
          else if (map[i][k] != '.')
               break;

    return count;
}
```