The King Hussein School for Computing Sciences
Department of Computer Science
Structured Programming - Spring 2022

Final Exam

Full Name: **Student ID:**

Question	Points	Score
1	16	
2	30	
3	40	
4	14	
Total	100	

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)

Saturday 18/6/2022

Question 1. Mini-Tracing (16 points) expected time = 5 - 15 minutes

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 that x, y and a are **global variables** defined as: `int x = 1, y = 2, a[3] = {5, 1, 9};`

	Code	Output

1.	`for (int i = 0; i < 3; i++)`
	`printf("%d", i);`
	`printf("%d", i+1);`
		Compilation Error

2.	`switch (x) {`
	`  case 1: printf("A");`
	`  case 2: printf("B");`
	`}`
		AB

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

4.	`printf("%d ", *(a+2));`	9
5.	`printf("%d ", *(++a));`	Compilation Error

6.	`int* ptr = &x;`
	`x++;`
	`printf("%d", *ptr);`
		2

7.	`void f(int c) {`
	`  int c = 5;`
	`  printf("%d", c);`
	`}`
	`int main() {`
	`  f(7);`
	`  return 0;`
	`}`
		Compilation Error

8.	`void fun(int* p1) {`
	`  (*p1)++;  printf("%d", *p1);`
	`}`
	`int main() {`
	`  fun(3);`
	`  return 0;`
	`}`
		Compilation Error

Question 2. Mini-Coding (30 points) expected time = 10 - 30 minutes

Implement each of the following functions.

(A)

```
// Returns 1 if a, b and c are the same
// or if they are all different.
// Returns 0 otherwise.
int zor(int a, int b, int c) {

    if (a == b && b == c)
        return 1;
    if (a != b && a != c && b != c)
        return 1;
    return 0;
}
```

(B)

```
// Returns the number of digits in x
int digits(int x) {
    int count = 1;
    while (x / 10  != 0) {
        count++;
        x /= 10;
    }
    return count;
}
```

(C)

```
// Returns 1 if all the integers in a[]
// are odd. Returns 0 otherwise.
int all_odd(int a[], int size) {

    for (int i = 0; i < size; i++)
        if (a[i] % 2 != 0)
            return 0;
    return 1;
}
```

(D)

```
// Fills (only) the first and last
// columns of the 20x20 array with 0
void fill_zeros(int a[20][20]) {

    for (int i = 0; i < 20; i++) {
       a[i][0] = 0;
       a[i][19] = 0;
    }
}
```

(E)

```
// Replaces all the characters of the given string with 'X'
void hide(char* str) {


    while (*str != '\0') {
       *str = 'X';
       str++;
    }
}
```

Grading Notes.

Each part of this question must be graded as follows:

- **6** points: completely correct.
- **4** points: 1 minor issue (e.g. off-by-one, forgetting to initialize a variable, fails at a corner case, etc.)
- **2** points: multiple minor issues.
- **0** points: Empty, more than 3 issues, or has one major issue.

Question 3. Code Writing (40 points)

expected time = 20 - 60 minutes

PART 1. ARRAYS AND LOOPS (14 points)

Implement a **void** function named **fill_random**(…) that receives a 1D array and its size as arguments and fills it with `100000` random integers in the range `[-100000, 100000]`, provided that **no number appears more than once** in the array.

SOLUTION 1

```
void fill_random(int a[], int size) {

      for (int i = 0; i < size; i++) {
          int random = -100000 +
                       rand() % 200001;
          int found = 0;
          for (int j = 0; j < i; j++)
              if (a[j] == random) {
                  found = 1;
                  break;
              }
          if (!found)
             i--;
          else
             a[i] = random;
      }
}
```

SOLUTION 2

```
void fill_random(int a[], int size) {
      int used[200001] = {0};

      for (int i = 0; i < size; i++) {
          int random = rand() % 200001;
          if (!used[random]) {
             a[i] = random - 100000;
             used[random] = 1;
          } else
             i--;
      }
}
```

PART 2. STRINGS (16 points)

Implement a program that reads two strings from the user and merges them into a new string, as the following examples show. The program should then print the resulting string.

Examples.

```
string 1 = "ccccc"
string 2 = "ggggg"
result   = "cgcgcgcgcg"

string 1 = "00000000"
string 2 = "TBA"
result   = "0T0B0A00000"

string 1 = "XYZ"
string 2 = "cccccc"
result   = "XcYcZcccc"

string 1 = ""
string 2 = "ABC"
result   = "ABC"
```

Notes.

- You can assume that no string entered by the user is longer than `100` characters.
- **Define your strings as arrays of characters.** However, you must use pointer arithmetic when processing the strings. You are not allowed to to use array notation anywhere other than when defining the strings.
- You are not allowed to use the `string.h` library.

```c
int main() {
	char str1[101];
	char str2[101];
	char result[2001];

	scanf("%s%s", str1, str2);

	char *ptr1 = str1, *ptr2 = str2, *ptr3 = result;
	while (*ptr1 != '\0' && *ptr2 != '\0') {
		*ptr3 = *ptr1;
		ptr1++; ptr3++;

		*ptr3 = *ptr2;
		ptr2++; ptr3++;
	}

	while (*ptr1 != '\0') {
		*ptr3 = *ptr1;
		ptr1++; ptr3++;
	}

	while (*ptr2 != '\0') {
		*ptr3 = *ptr2;
		ptr2++; ptr3++;
	}

	*ptr3 = '\0';
	printf("%s", result);
	return 0;
}
```

PART 3. RECURSION (10 points)

Implement a **recursive** function named **`plot(...)`** that receives as arguments an array and its size, and plots a histogram of the integers that are in the array (see the example provided below).

Notes.

- You are allowed to use loops *with recursion*. You are **not** allowed to use loops *without* recursion.
- You are allowed to add extra parameters to the function if you need, provided that you explain in a comment what the parameters are.

Example.

Array: `a[] = {1, 5, 0, 3, 7}`

Output:

```
1: *
5: *****
0:
3: ***
7: *******
```

```
void plot(int a[], int size, int i) {
    if (i >= size || i < 0)
        return;

    printf("%d: ", a[i]);
    for (int j = 0; j < a[i]; j++)
        printf("*");
    printf("\n");

    plot(a, size, i+1);
}
```

Question 4. Code Reading (14 points)

expected time = 5 - 15 minutes

PART 1. What is the output of the function below in each of the following cases?

```
void mystery(int N) {
    int a, b, c = 0;

    for (int i = 0; i < N; i++) {
        scanf("%d", &a);
        if (a % 2 == 0) {
            b = a;
            c++;
        }
    }

    if (c > 0)
        printf("%d", b);
}
```

A. If `N = 2` and the user input is `4 7`

`4` (2 points)

B. If `N = 3` and the user input is `1 1 1`

`No output` (2 points)

C. If `N = 100` and the user input is `1 2 3 4 … 100`

`100` (2 points)

What is the purpose of (الهدف من) this code?

Do not use > 20 words in your answer (write in the box).

```
If there are even numbers in the input,
    the last even number is printed
```

PART 2. What is the content of the array `a[][]` after executing **`mystery`** in each of the following cases?

```
// assume N is a global constant
// defined and initialized here

void mystery(int a[N][N]) {
    for (int i = 0; i < N; i++) {
        int temp = a[i][i];
        a[i][i] = a[i][N-i-1];
        a[i][N-i-1] = temp;
    }
}
```

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

`{1}` (2 points)

B. If `N=2` and `a = {{1, 2},`
`{3, 4}}`

```
{{2, 1}                (3 points)
 {4, 3}}
```

C. If `N=100` and every row in the array contains `{1, 2, 3, 4, 5, …, 97, 98, 99, 100}`

show only the last 3 elements of the last 3 rows.

```
{… 3,  99, 100},          (3 points)
{… 98, 2,  100},
{… 98, 99, 1}
```

What is the purpose of (الهدف من) this code?

Do not use > 20 words in your answer (write in the box).

```
Swaps the main diagonals of the array
```

This page is intentionally left blank