

## Domenic Iorfida

1. What does the following piece of code output for  $n = 7$ ? (3 marks)

```
int fun1(int n)
{
    if (n == 1){
        return 0;
    }
    else {
        return (1 + fun1(n / 2));
    }
}
```

n	return
7	2
3	1
1	0

Output: 2

2. What is the purpose of a base-case in a recursive function? What happens if no base-case is provided? (2 marks (1 + 1))

The base case acts as an escape route. If there was no base case, the function would repeat infinitely and a stack overflow error would occur.

3. What is the base-case for the following piece of code? What will be the output for  $n = 10$ ? (3 marks (1 + 2))

```
void fun2(int n)
{
    if(n == 0) {
        return;
    }
    fun2(n/2);
    printf("%d ", n%2);
}
```

Base case:  $n = 0$

n	output printed
10	0
5	1
2	0
1	1

0	(returned, nothing printed)
---	-----------------------------

Output: 1 0 1 0

4. What are the similarities and differences between iteration and recursion? When is it wise to use recursive functions instead of iterations? (6 marks (3 + 3))

Similarities: both can be used for similar tasks; both involve checking a condition and performing tasks based on said condition.

Differences: iteration uses a counter while recursion uses stacks; recursion has a “base case” while iteration only has a conditional statement.

Recursion is great in specific cases where a larger problem can be broken down into little chunks and evaluated that way. It would be way easier than an iterative function.

5. Consider the following program, provide the output and provide the content of the function stack from start to end. (10 marks (4 + 6))

```
#include<stdio.h>
void abc(int i);
int main()
{
    abc(12);
    return 0;
}
void abc(int i)
{
    if(i == 15) {
        return;
    }
    abc(i + 1);
    printf("%d ", i);
}
```

Function stack:

prints...	i
12 (exits the function)	12
13 (returns control to 12)	13
14 (returns control to 13)	14
(returns control to 14)	15

Output: 14 13 12

---

6. Indicate if the following questions are true or false: (4 marks)

- F a. Recursive functions must return a value.
  - F b. Not all recursive definitions may be written iteratively.
  - T c. There can be more than one stopping case in a recursive function.
  - T d. the main( ) function can be called recursively.
- 

7. (12 marks)

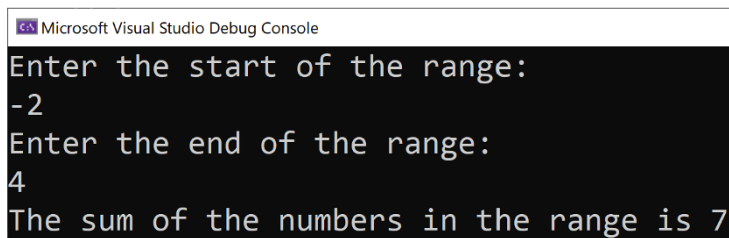
- Provide your source code (.c) file (as separate files). **Do not zip your work!**
- Provide snapshots of **all** your results after running your code. Use a word or pdf file to show your results.
- Provide comments in your code.

Write a C program that has a **recursive** function called *sumOfRange*. The function has two input integer parameters start and end. The function should return the sum of numbers started from the start number to the end number. Start and end numbers should be included in the sum value. The function has the following prototype:

**int sumOfRange(int start, int end);**

Create a complete program to test the function. In the main function scan start and end values, call *sumOfRange* function, and print the result.

Here is a sample run:



```
Microsoft Visual Studio Debug Console
Enter the start of the range:
-2
Enter the end of the range:
4
The sum of the numbers in the range is 7
```

### Code Output Screenshots:

```
❖ make -s
❖ ./main
Enter the start of the range:
2
Enter the end of the range:
10
The sum of the numbers in the range is 54❖
```

```
❖ make -s
❖ ./main
Enter the start of the range:
-5
Enter the end of the range:
5
The sum of the numbers in the range is 0❖
```

```
❖ make -s
❖ ./main
Enter the start of the range:
27
Enter the end of the range:
7
The start value must be less than end value

Enter the start of the range:
7
Enter the end of the range:
27
The sum of the numbers in the range is 357❖
```