Assignment-4 Part-B: Functional Programming

Total Points = 50

Instructions:

1. You must use ML language (**SML**) to implement these functional programming programs.
2. You may optionally use the online version of SML here.
3. You must submit this assignment on Canvas by clicking the title link for the assignment. Make sure your programs compile (without any compiler errors). You will not receive credit if your program for a problem does not compile. If you are unable to complete any of the programs, submit the parts that work (with no compiler errors) for partial credit.
4. Your grade will be based on meeting the requirements, functionality (does the program do what it is supposed to do), readability (is the code nicely indented and formatted), and understandability (are the literals meaningful and is the code well documented with appropriate comments). You must incorporate all the good programming practices and styles.

Deliverables:

1. You should **submit one PDF file**, named **assign4B.pdf**. This PDF file should contain **code to the solution** of the problems (in text format only, no snapshot will be accepted) and **your program's sample runs** with output (**in text format only, no snapshot will be accepted**). Make sure to paste your code correctly (with correct indentation). If you have any text for the grader to read before grading, you can include it at the top of this file as comments or in blackboard notes.
2. You should **also submit one sml file** named **assign4B.sml**. This file should contain your code for the individual problems(s) in the assignment for the grader to run. You should separate your code by adding comments for each problem.

1.

A) Sum of a List: **(5 Points)**

Write a function **sumList (L)** that returns the sum of the numbers in the list **L**.

Sample Run:

```
- sumList [22,4,7,1];
```

Output:

```
val it = 34 : int
```

B) Fibonacci: **(5 Points)**

Write a function **fibonacci (n)** that computes the nth Fibonacci number for a given number **n**. For example: *fibonacci (6)* will give us *8*. The Fibonacci sequence is a set of numbers that starts with a one or a zero, followed by a one, and proceeds based on the rule that each number (called a Fibonacci number) is equal to the sum of the preceding two numbers.

Sample Run:

```
- fibonacci (6);
```

Output:

```
val it = 8 : int
```

---

2. Reverse (Tail Recursive): **(10 Points)**

Write a tail recursive function reverse(L) that returns a list that is the reverse of the list L. The top-level call is as follows:

reverse(L) returns a list that is the reverse of **L**

Sample Run:

```
- reverse([1, 2, 3, 4, 5]);
```

Output:

```
val it = [5, 4, 3, 2, 1] : int
```

---

3. **Rotate:** **(10 Points)**

Write a function **rotate(L:int L, n:int)** that takes the list **L** and an integer as input, and performs a circular shift of the elements in the list in clockwise direction by **n** positions. Your program should handle the case where **n** is larger than the length of the list **L**.

Sample Run 1:

- *rotate([1, 2, 3, 4, 5], 3);*

Output 1:

val it = [3, 4, 5, 1, 2] : int

Sample Run 2:

- *rotate([1, 2, 3, 4, 5], 1);*

Output 2:

val it = [5, 1, 2, 3, 4] : int

---

4. **Merge Sort:** **(10 Points)**

Write a program for mergesort in ML. Top level call should be **msort(L)** which will return the list **L** sorted in ascending order.

Sample Run:

- *msort([2, 5, 3, 4, 1]);*

Output:

val it = [1, 2, 3, 4, 5] : int

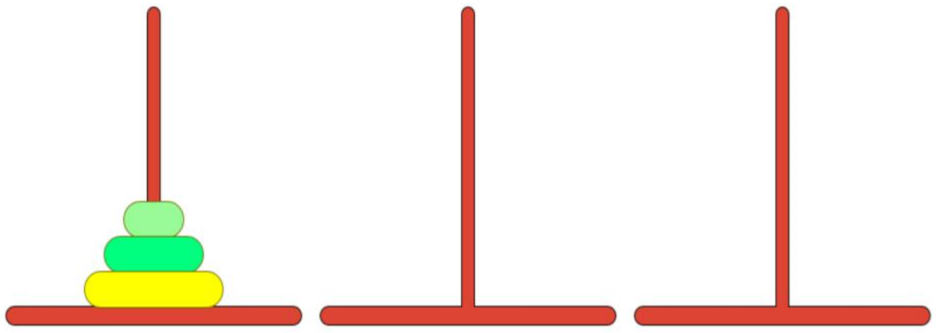
---

5. Tower of Hanoi: **(10 Points)**

Program the Tower of Hanoi puzzle. Your top-level call should be of the form:

hanoi(n, peg1, peg2, peg3)

where n is the number of disks on peg1, and peg1, peg2, peg3 are the parameters holding the names of the 3 pegs. hanoi(n, peg1, peg2, peg3) returns a list of moves where each move is a pair.



Sample Run:

- *hanoi(3, 1, 2, 3);*

Output:

```
val it = [(1,3), (1,2), (3,2), (1,3), (2,1), (2,3), (1,3)] : (int * int) list
```