Assignment 11

一、概念题

1、什么是函数式编程？它有什么优缺点。

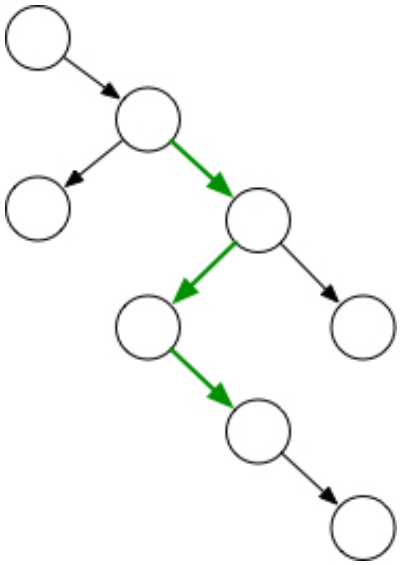
2、什么是尾递归和尾递归优化?

3、C++中的Filter、Map和Reduce操作各自是什么含义。

4、C++是如何实现Currying操作的？并阐述一下该操作的重要性。

二、编程题

1、使用尾部递归实现算法来找到二叉树中最长的Z型路径长度。



如图，该二叉树的最长Z型路径长度为3。

> https://leetcode-cn.com/problems/longest-zigzag-path-in-a-binary-tree/

2、函数求导数（currying）。

现有一系列目标函数如sin、cos等等，我们需要对它们求导。

1. 你首先需要实现求导数函数derivative。
2. 接着实现bind_derivative函数对该函数进行柯里化，生成对固定点求导数的函数d1和d2。
3. 最终利用这些函数对目标函数求导。

具体细节请阅读以下代码：

```cpp
#include <vector>
#include <functional>
#include <cmath>
/* 求导数函数，对某个函数f在点x0处求得导数
 * f'(x0) = (f(x0)-f(x0-d))/d
 * params:
 * x: x0
 * d: d
 * f: f
 */
double derivative(double x, double d, double (*f)(double)) {
  ...
}

int main() {
  std::vector<double (*)(double)> funcs = {sin, cos, tan, exp, sqrt, log, log10};
// 目标函数
  auto d1 = bind_derivative(1, 0.000001); // 在x=1处求导数的函数d1
  auto d2 = bind_derivative(1)(0.000001); // 在x=1处求导数的函数d2
  std::vector<double> result1, result2;
  std::transform(funcs.begin(), funcs.end(), std::back_inserter(result1), d1);
  std::transform(funcs.begin(), funcs.end(), std::back_inserter(result2), d2);
  // result1的结果与result2的结果相同
  return 0;
}
```

要求：利用std::bind实现**bind_derivative**函数达到示例代码中的效果。

提示：可以结合lambda函数实现。