```
Utku Kurtulmuş 21903025 EEE448
Q1) Solution Implemented in c++:
#include <iostream>
#include <unordered_map>
using namespace std;
//memory for dynamic programming
unordered map<int, long long> memory; // Memoization to store computed values
//fibonacci recursive
long long recursive(int n) {
  if(n \le 1)
    return n ? 1:0;
  }
  return recursive(n - 1) + recursive (n - 2);
}
//fibonacci iterative
long long iterative(int n) {
  long long previous = 0;
  long long current = 1;
  long long dummy;
  if(n == 0) {
    return 0;
  }
  for(int i = 2; i \le n; i++) {
    dummy = current;
    current = current + previous;
    previous = dummy;
  }
  return current;
}
//fibonacci dynamic programming (top-down)
long long dynamic(int n) {
  //if the value is in memory return it
  if (memory.find(n) != memory.end()) {
    return memory[n];
  //base case
  if(n \le 1)
    return n? 1:0;
```

}

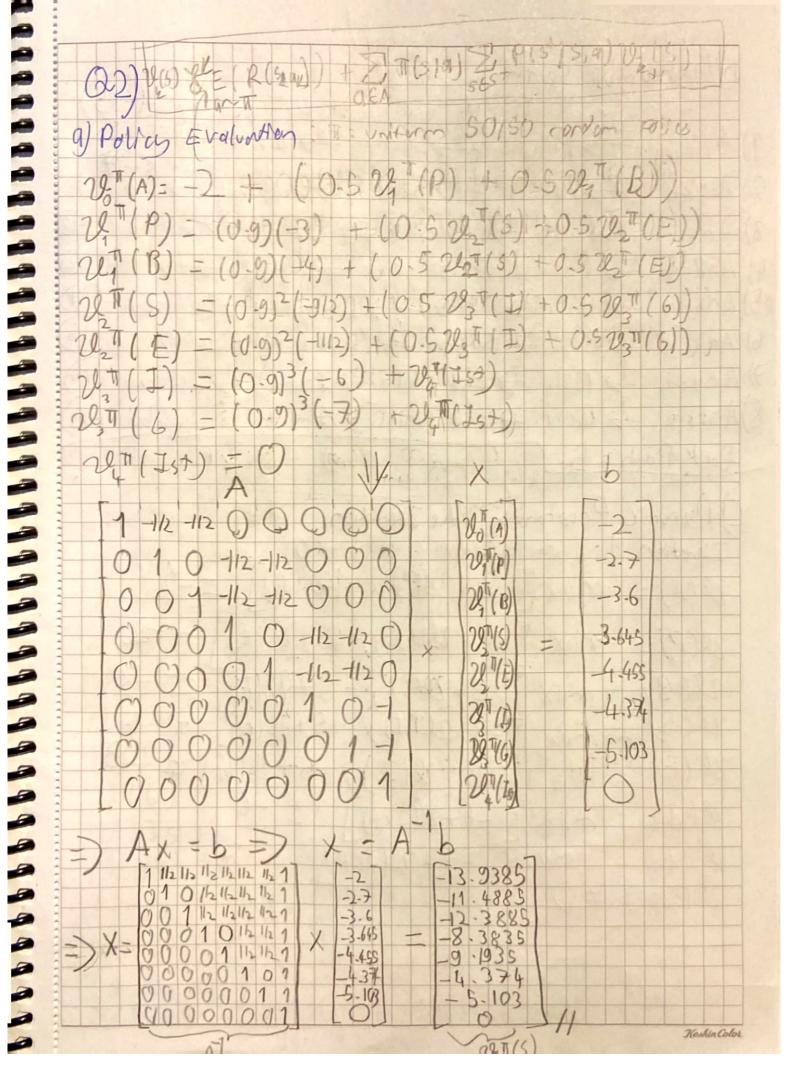
```
// If the value is not in memory, compute it and store it in memory for future use
  memory[n] = dynamic(n - 1) + dynamic(n - 2);
  return memory[n];
}
int main() {
  iterative(45);
  //dynamic(45);
  //recursive(45);
  /*
  //for test purposes
  //recursive
  cout << "Recursive" << endl;
  for(int i = 0; i < 10; i++) {
     cout << recursive(i) << endl;</pre>
  }
  //iterative
  cout << "Iterative" << endl;</pre>
  for(int i = 0; i < 10; i++) {
     cout << iterative(i) << endl;</pre>
  }
  //dynamic programming
  cout << "Dynamic" << endl;</pre>
  for(int i = 0; i < 10; i++) {
     cout << dynamic(i) << endl;</pre>
  }
  */
  return 0;
}
```

Test Results:

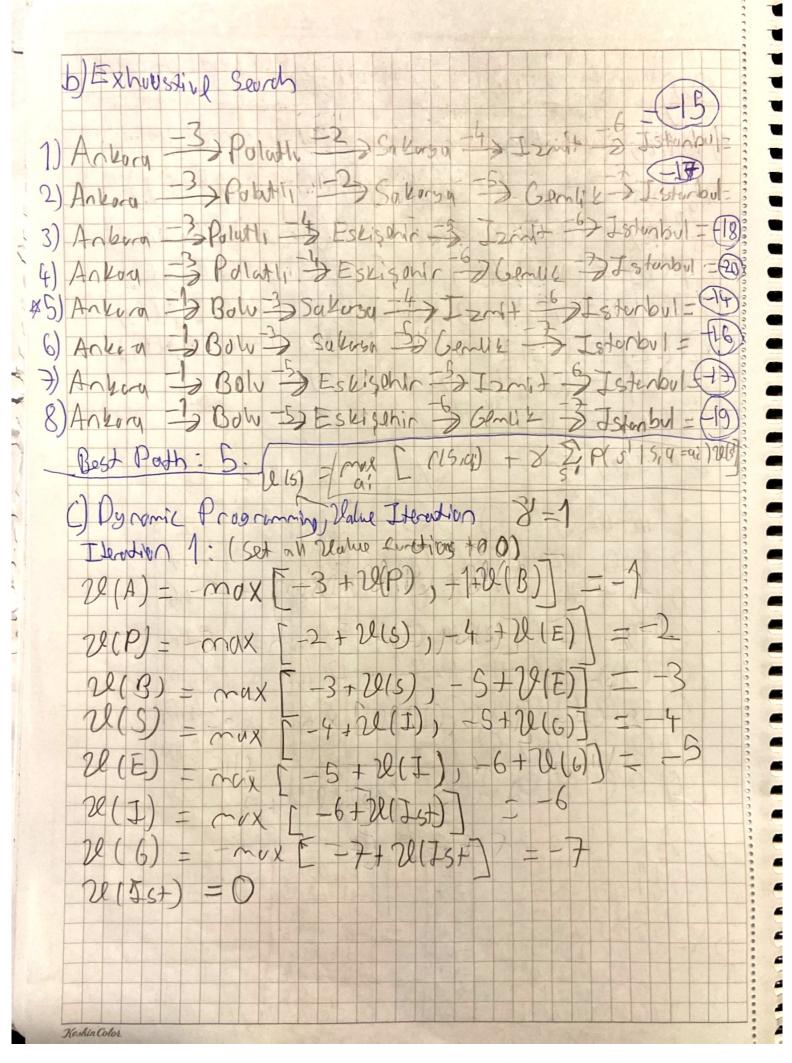
For 45th fibonacci number

Iterative: 0.044s Dynamic: 0.045s Recursive: 6.529s

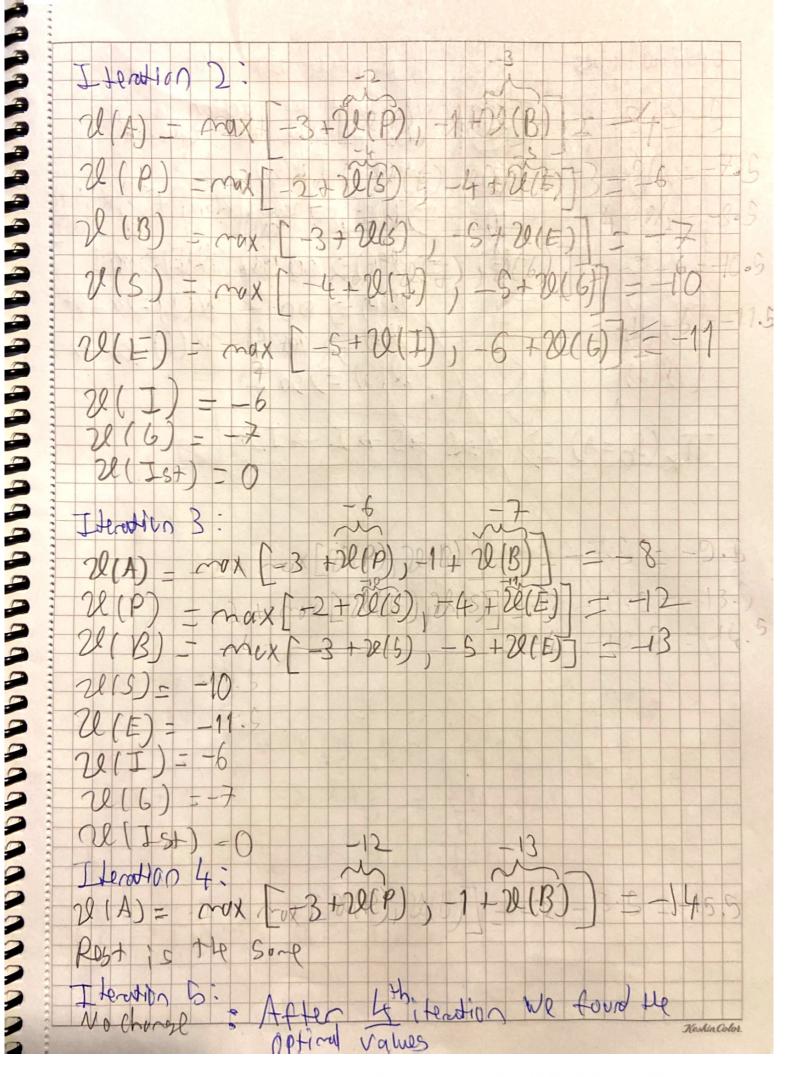
Recursive one is horrible as expected since the time complexity is O(2^n)



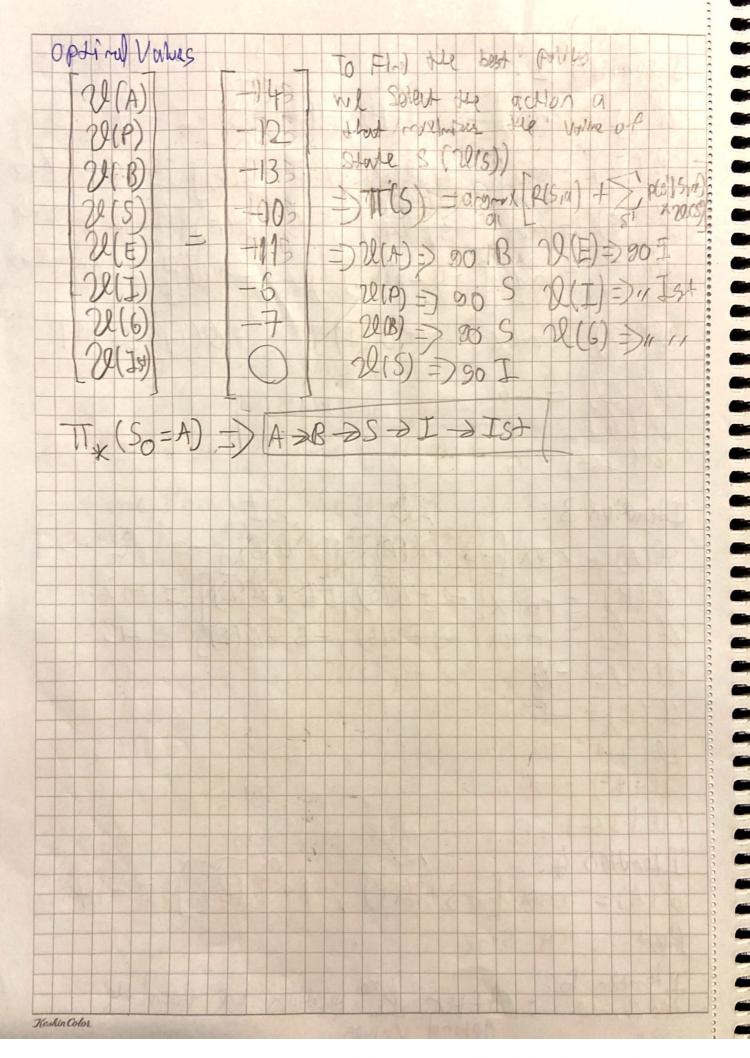
Scanned with CamScanner



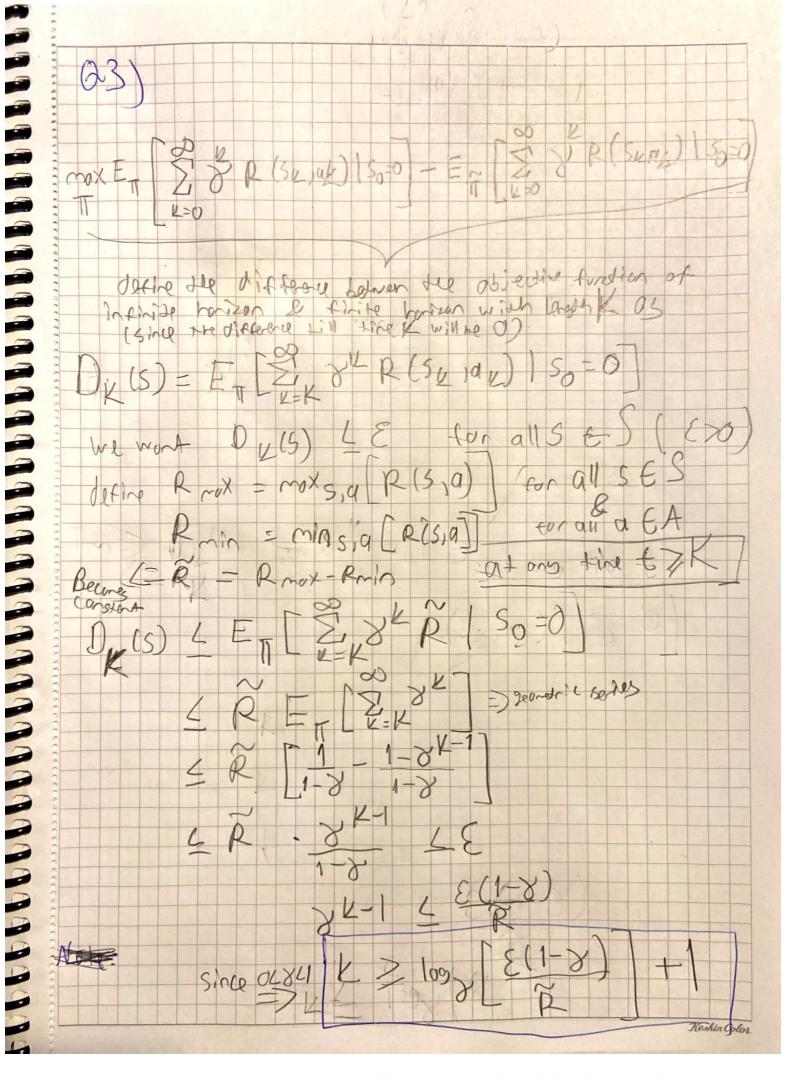
Scanned with CamScanner



Scanned with CamScanner



Scanned with CamScanner



Scanned with CamScanner