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 <= 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 <= 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 <= 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;
    }

    //iterative
    cout << "Iterative" << endl;
    for(int i = 0; i < 10; i++) {
        cout << iterative(i) << endl;
    }

    //dynamic programming
    cout << "Dynamic" << endl;
    for(int i = 0; i < 10; i++) {
        cout << dynamic(i) << endl;
    }
    */

    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)

Q2) $v_k(s) = \mathbb{E}_{a \sim \pi}\left(R(s_k, a_k) \right) + \sum_{a \in A} \pi(s|a) \sum_{s \in S^+} P(s' | s, a) \, v_{k+1}(s)$

a) Policy Evaluation : π : uniform 50/50 random policy

$$v_0^{\pi}(A) = -2 + \left(0.5 v_1^{\pi}(P) + 0.5 v_1^{\pi}(B)\right)$$
$$v_1^{\pi}(P) = (0.9)(-3) + \left(0.5 v_2^{\pi}(S) + 0.5 v_2^{\pi}(E)\right)$$
$$v_1^{\pi}(B) = (0.9)(-4) + \left(0.5 v_2^{\pi}(S) + 0.5 v_2^{\pi}(E)\right)$$
$$v_2^{\pi}(S) = (0.9)^2(-9/2) + \left(0.5 v_3^{\pi}(I) + 0.5 v_3^{\pi}(G)\right)$$
$$v_2^{\pi}(E) = (0.9)^2(-11/2) + \left(0.5 v_3^{\pi}(I) + 0.5 v_3^{\pi}(G)\right)$$
$$v_3^{\pi}(I) = (0.9)^3(-6) + v_4^{\pi}(I_{st})$$
$$v_3^{\pi}(G) = (0.9)^3(-7) + v_4^{\pi}(I_{st})$$
$$v_4^{\pi}(I_{st}) = 0$$

$$\underbrace{\begin{bmatrix} 1 & -1/2 & -1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1/2 & -1/2 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1/2 & -1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1/2 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1/2 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} v_0^{\pi}(A) \\ v_1^{\pi}(P) \\ v_1^{\pi}(B) \\ v_2^{\pi}(S) \\ v_2^{\pi}(E) \\ v_3^{\pi}(I) \\ v_3^{\pi}(G) \\ v_4^{\pi}(I_{st}) \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} -2 \\ -2.7 \\ -3.6 \\ -3.645 \\ -4.455 \\ -4.374 \\ -5.103 \\ 0 \end{bmatrix}}_{b}$$

$$\Rightarrow Ax = b \Rightarrow x = A^{-1} b$$

$$\Rightarrow X = \underbrace{\begin{bmatrix} 1 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1 \\ 0 & 1 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 1 \\ 0 & 0 & 1 & 1/2 & 1/2 & 1/2 & 1/2 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1/2 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1/2 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A^{-1}} \times \begin{bmatrix} -2 \\ -2.7 \\ -3.6 \\ -3.645 \\ -4.455 \\ -4.374 \\ -5.103 \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} -13.9385 \\ -11.4885 \\ -12.3885 \\ -8.3835 \\ -9.1935 \\ -4.374 \\ -5.103 \\ 0 \end{bmatrix}}_{v^{\pi}(s)} //$$

b) Exhaustive Search

1) Ankara $\xrightarrow{-3}$ Polatlı $\xrightarrow{-2}$ Sakarya $\xrightarrow{-4}$ İzmit $\xrightarrow{-6}$ İstanbul = (-15)
2) Ankara $\xrightarrow{-3}$ Polatlı $\xrightarrow{-2}$ Sakarya $\xrightarrow{-5}$ Gemlik $\xrightarrow{-7}$ İstanbul = (-17)
3) Ankara $\xrightarrow{-3}$ Polatlı $\xrightarrow{-4}$ Eskişehir $\xrightarrow{-5}$ İzmit $\xrightarrow{-6}$ İstanbul = (-18)
4) Ankara $\xrightarrow{-3}$ Polatlı $\xrightarrow{-4}$ Eskişehir $\xrightarrow{-6}$ Gemlik $\xrightarrow{-7}$ İstanbul = (-20)
*5) Ankara $\xrightarrow{-1}$ Bolu $\xrightarrow{-3}$ Sakarya $\xrightarrow{-4}$ İzmit $\xrightarrow{-6}$ İstanbul = (-14)
6) Ankara $\xrightarrow{-1}$ Bolu $\xrightarrow{-3}$ Sakarya $\xrightarrow{-5}$ Gemlik $\xrightarrow{-7}$ İstanbul = (-16)
7) Ankara $\xrightarrow{-1}$ Bolu $\xrightarrow{-5}$ Eskişehir $\xrightarrow{-5}$ İzmit $\xrightarrow{-6}$ İstanbul = (-17)
8) Ankara $\xrightarrow{-1}$ Bolu $\xrightarrow{-5}$ Eskişehir $\xrightarrow{-6}$ Gemlik $\xrightarrow{-7}$ İstanbul = (-19)

Best Path: 5.

$$v(s) = \max_{a_i} \left[r(s,a_i) + \gamma \sum_{s'} P(s' \mid s, a=a_i) v(s') \right]$$

c) Dynamic Programming, Value Iteration $\gamma = 1$

Iteration 1: (Set all value functions to 0)

$v(A) = \max[-3 + v(P), -1 + v(B)] = -1$
$v(P) = \max[-2 + v(S), -4 + v(E)] = -2$
$v(B) = \max[-3 + v(S), -5 + v(E)] = -3$
$v(S) = \max[-4 + v(I), -5 + v(G)] = -4$
$v(E) = \max[-5 + v(I), -6 + v(G)] = -5$
$v(I) = \max[-6 + v(Ist)] = -6$
$v(G) = \max[-7 + v(Ist)] = -7$
$v(Ist) = 0$

Iteration 2:

$v(A) = \max[-3 + v(P), -1 + v(B)] = -4$ (-2, -3)

$v(P) = \max[-2 + v(S), -4 + v(E)] = -6$ (-4, -5)

$v(B) = \max[-3 + v(S), -5 + v(E)] = -7$

$v(S) = \max[-4 + v(I), -5 + v(G)] = -10$

$v(E) = \max[-5 + v(I), -6 + v(G)] = -11$

$v(I) = -6$

$v(G) = -7$

$v(Ist) = 0$

Iteration 3:

$v(A) = \max[-3 + v(P), -1 + v(B)] = -8$ (-6, -7)

$v(P) = \max[-2 + v(S), -4 + v(E)] = -12$ (-10, -11)

$v(B) = \max[-3 + v(S), -5 + v(E)] = -13$

$v(S) = -10$

$v(E) = -11$

$v(I) = -6$

$v(G) = -7$

$v(Ist) = 0$

Iteration 4:

$v(A) = \max[-3 + v(P), -1 + v(B)] = -14$ (-12, -13)

Rest is the same

Iteration 5: No change : After 4th iteration we found the optimal values

Optimal Values

$$\begin{bmatrix} v(A) \\ v(P) \\ v(B) \\ v(S) \\ v(E) \\ v(I) \\ v(G) \\ v(I_{s+}) \end{bmatrix} = \begin{bmatrix} -14 \\ -12 \\ -13 \\ -10 \\ -11 \\ -6 \\ -7 \\ 0 \end{bmatrix}$$

To Find the best policy we select the action a that maximizes the value of state S ($v(s)$)

$$\Rightarrow \pi(S) = \arg\max_a \left[R(S,a) + \sum_{S'} P(S'|S,a) * v(S') \right]$$

$\Rightarrow v(A) \Rightarrow$ go B $\quad v(E) \Rightarrow$ go I

$v(P) \Rightarrow$ go S $\quad v(I) \Rightarrow$ " I_{s+}

$v(B) \Rightarrow$ go S $\quad v(G) \Rightarrow$ " "

$v(S) \Rightarrow$ go I

$\pi_*(S_0 = A) \Rightarrow$ **A → B → S → I → I_{s+}**

Keshin Color

Q3)

$$\max_{\pi} E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k) \mid s_0 = 0\right] - E_{\tilde{\pi}}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k) \mid s_0 = 0\right]$$

define the difference between the objective function of infinite horizon & finite horizon with length K as (since the difference till time K will be 0)

$$D_K(s) = E_{\pi}\left[\sum_{k=K}^{\infty} \gamma^k R(s_k, a_k) \mid s_0 = 0\right]$$

we want $D_k(s) \leq \varepsilon$ for all $s \in S$ $(\varepsilon > 0)$

define $R_{max} = \max_{s,a}[R(s,a)]$ for all $s \in S$ & for all $a \in A$ at any time $t \geq K$

$R_{min} = \min_{s,a}[R(s,a)]$

$\leq \tilde{R} = R_{max} - R_{min}$ (becomes constant)

$$D_K(s) \leq E_{\pi}\left[\sum_{k=K}^{\infty} \gamma^k \tilde{R} \mid s_0 = 0\right]$$

$$\leq \tilde{R}\, E_{\pi}\left[\sum_{k=K}^{\infty} \gamma^k\right] \Rightarrow \text{geometric series}$$

$$\leq \tilde{R}\left[\frac{1}{1-\gamma} - \frac{1-\gamma^{K-1}}{1-\gamma}\right]$$

$$\leq \tilde{R} \cdot \frac{\gamma^{K-1}}{1-\gamma} \leq \varepsilon$$

$$\gamma^{K-1} \leq \frac{\varepsilon(1-\gamma)}{\tilde{R}}$$

since $0 < \gamma < 1$ $\Rightarrow K$

$$K \geq \log_{\gamma}\left[\frac{\varepsilon(1-\gamma)}{\tilde{R}}\right] + 1$$