DP FORM 1

Q1. Given a string that consists of A/B/C/D but some of the places are missing and denoted by '?' Find :

(a) Find no. of ways to fill the missing places (?) with A/B/C/D such that no two neighbours are same.

(b) What if the string is circular. What if the no of A's has to be odd.

(c) Print lexicographically shortest solution.

S = "??A?B?D?"

Restrictions :-

1. No two neighbours are the same
2. The string is ~~cir~~ circular
3. The no. of A's has to be odd.

Sol

- Form 1
- State

DP(i, prev, first (#A)%2) $\rightarrow$ No. of ways to fill '?' in [i..N]

prev - previous value. This ensure no two neighbours are same

first - value at 0th index. This is to ensure our string is circular.

(#A%2) - stores the parity (0 for even & 1 for odd) of the no. of A's placed so far. This ensures 3rd restriction.

- Transition

$$\begin{tikzcd}
 & \sum DP(i+1, A, first, (cnta+1)\%2) \\
DP(i, prev, first, cnta) \arrow[ur, "A"] \arrow[r, "B"] \arrow[dr, "X", "i==0"'] \arrow[d, "i==N"] & DP(i+1, B, first, cnta) \\
\text{if (prev != first \&\& cnta==1) return 1; else return 0.} & DP(i+1, \cancel{x}, X, \ldots)
\end{tikzcd}$$

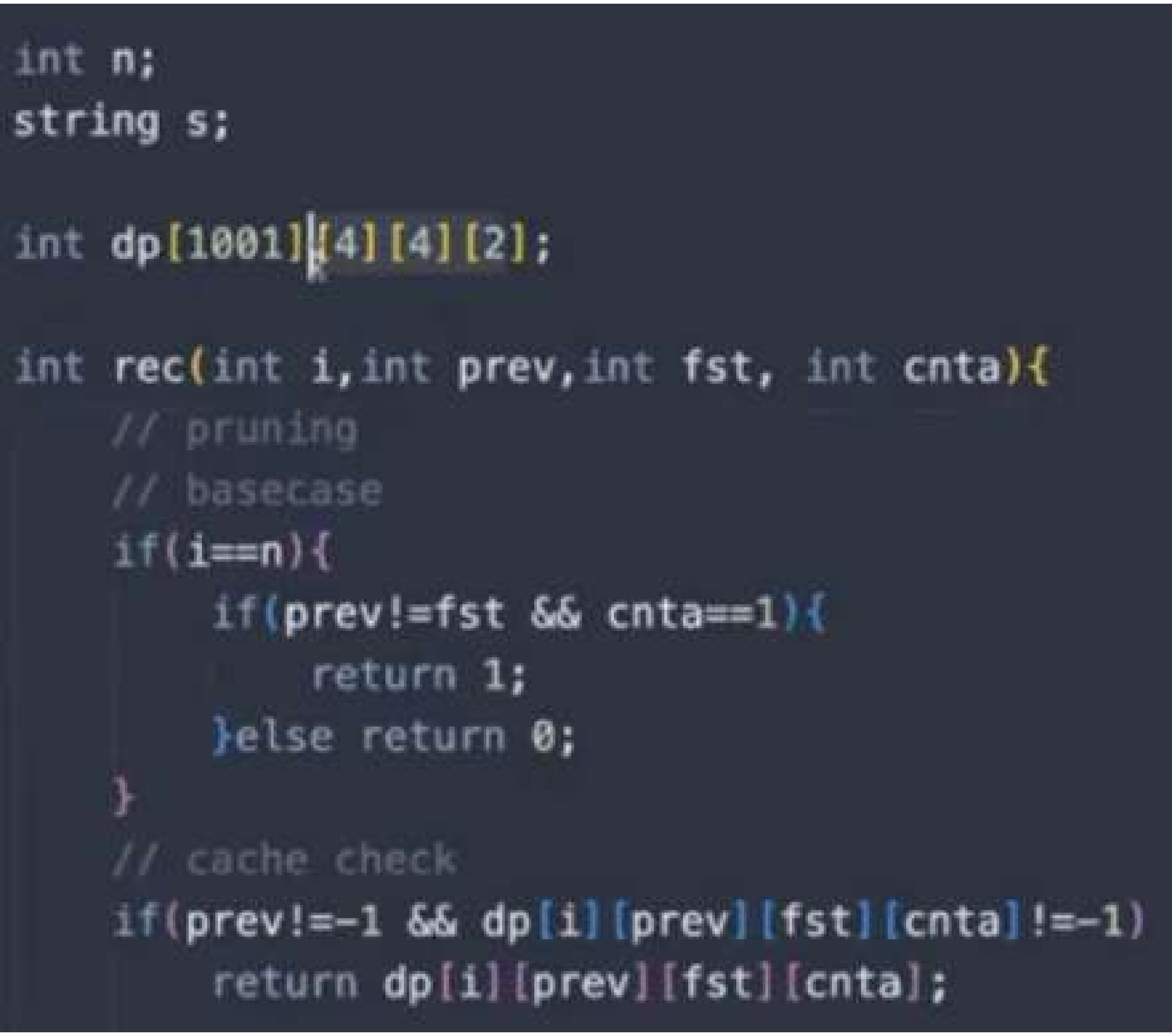
- TLE check

# of state - N×4×4×2

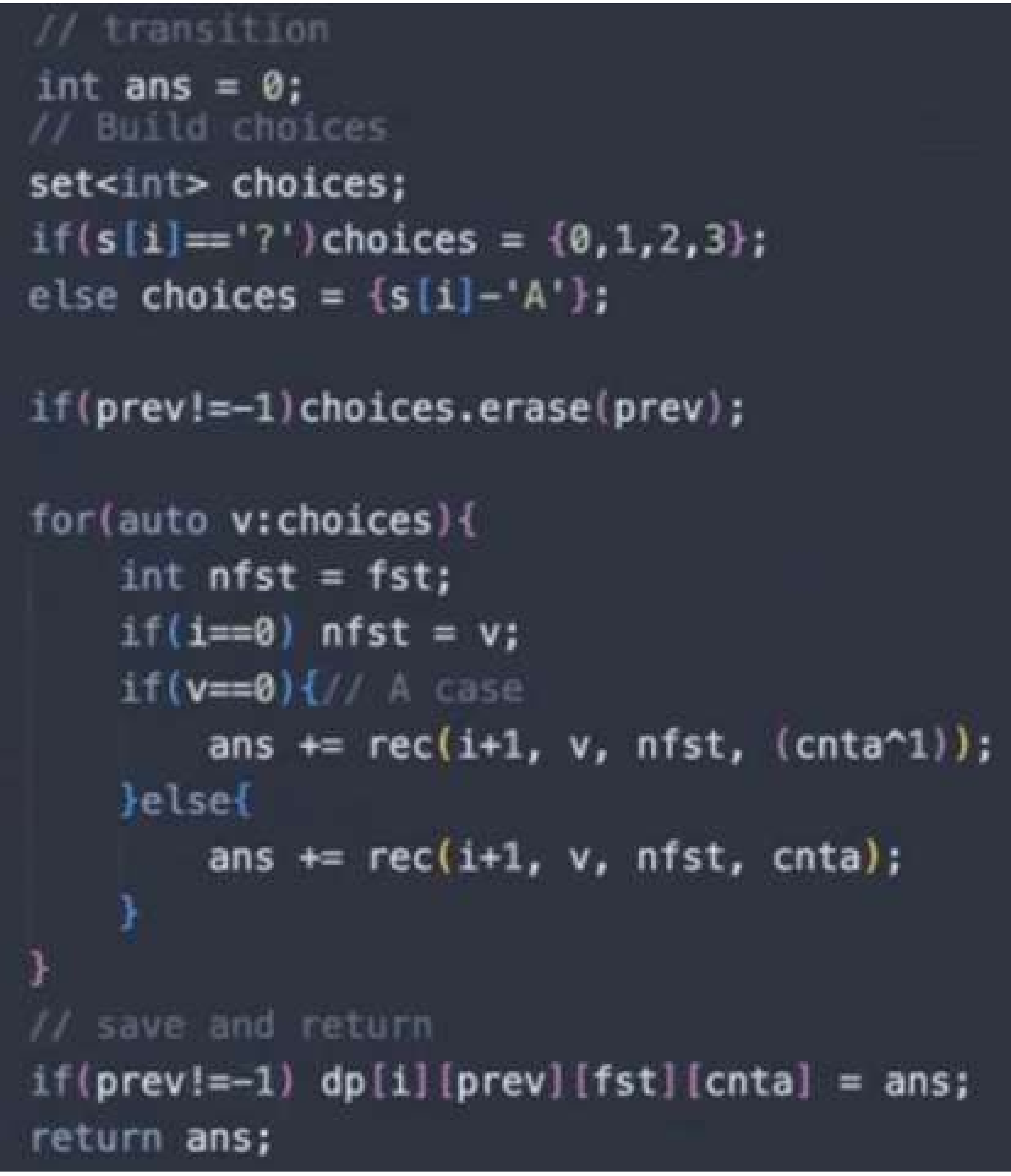
$DP(\underset{N}{i}, \underset{\substack{4\\(A/B/C/D)}}{prev}, \underset{4\,(A/B/C/D}{first}, \underset{2(0,1)}{cnt\,a\%2})$

# of Transition = 4

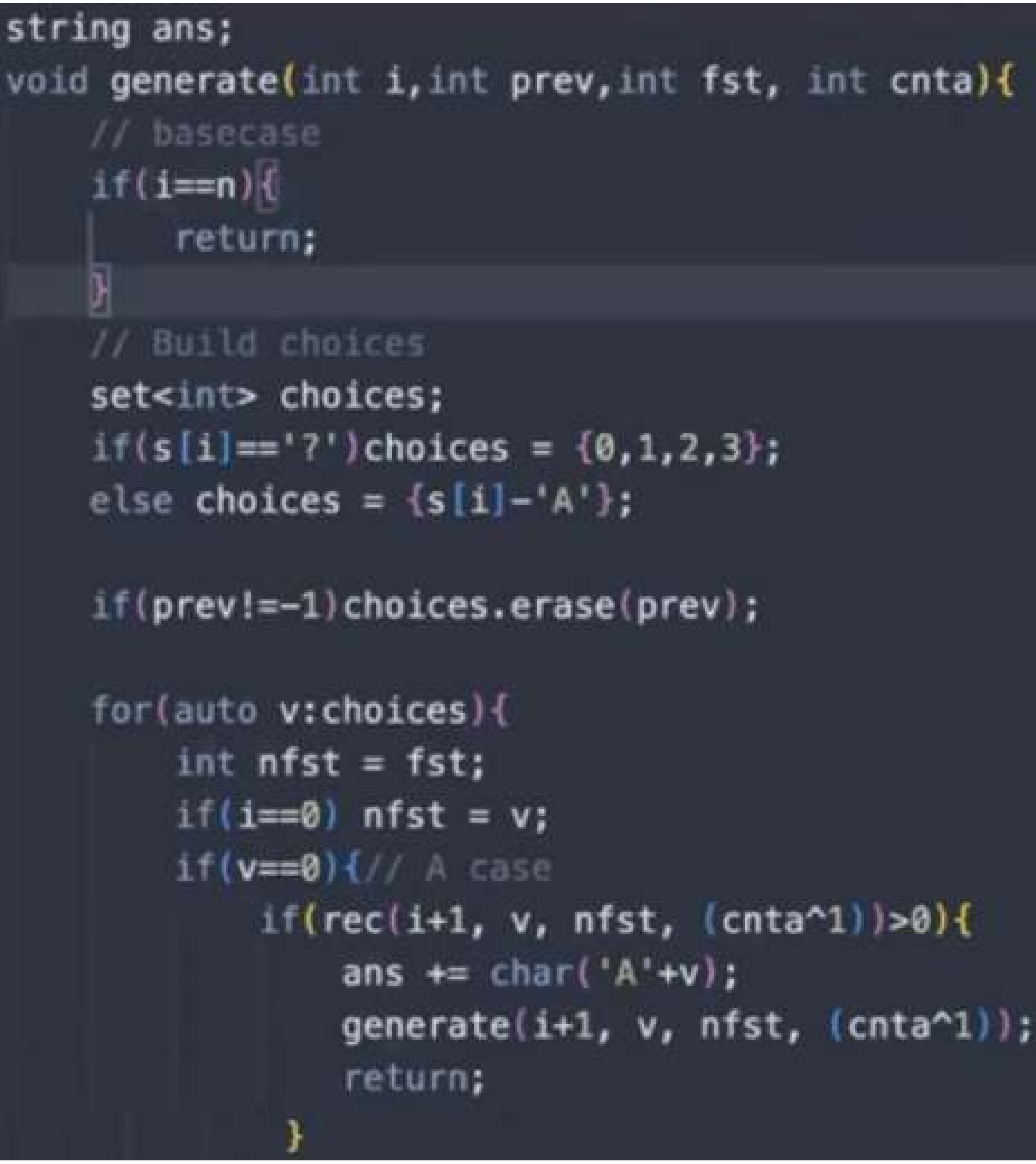
T.C = O(32N(1+4)) = O(N)



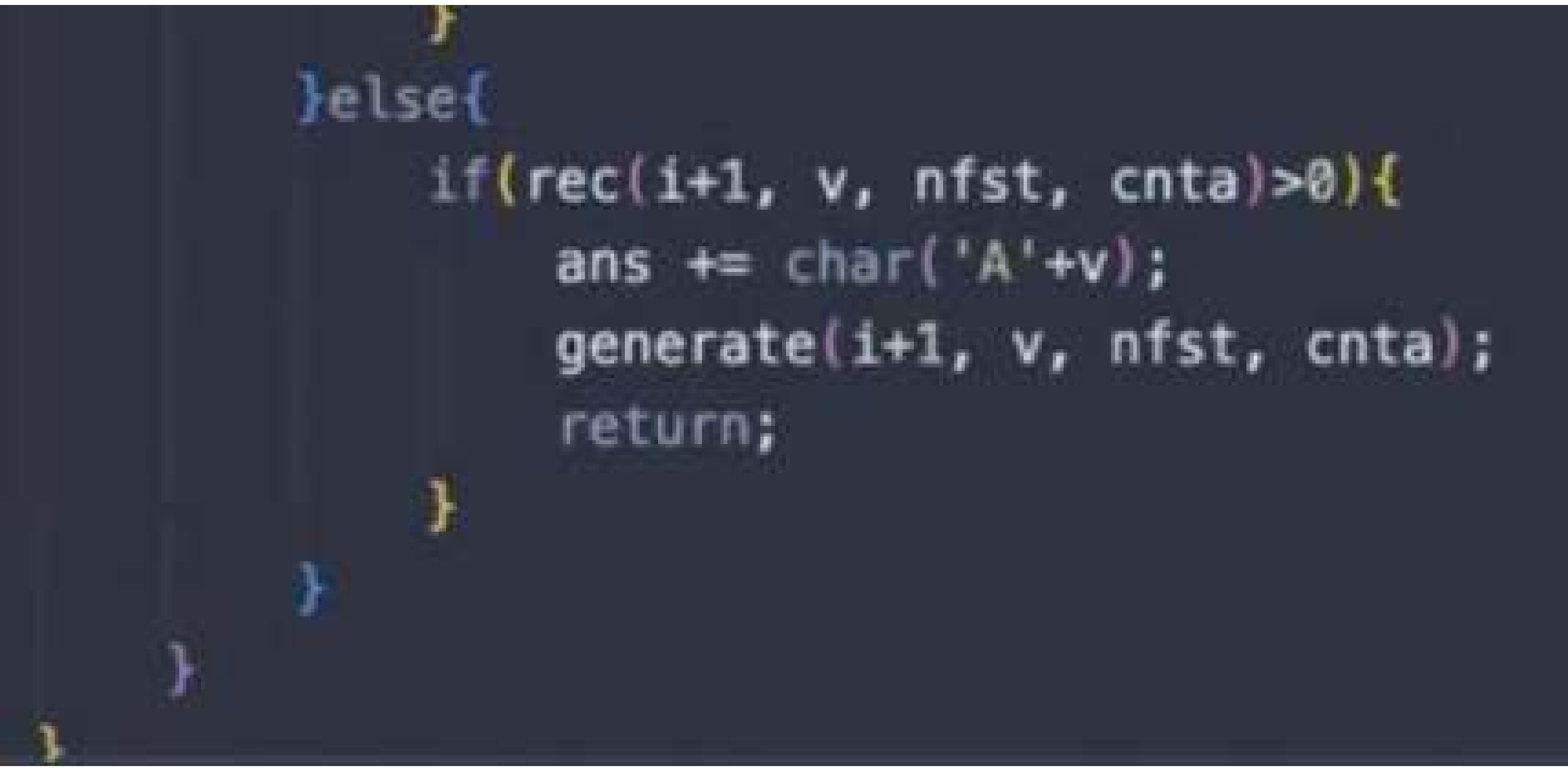
```
int n;
string s;

int dp[1001][4][4][2];

int rec(int i,int prev,int fst, int cnta){
    // pruning
    // basecase
    if(i==n){
        if(prev!=fst && cnta==1){
            return 1;
        }else return 0;
    }
    // cache check
    if(prev!=-1 && dp[i][prev][fst][cnta]!=-1)
        return dp[i][prev][fst][cnta];
```



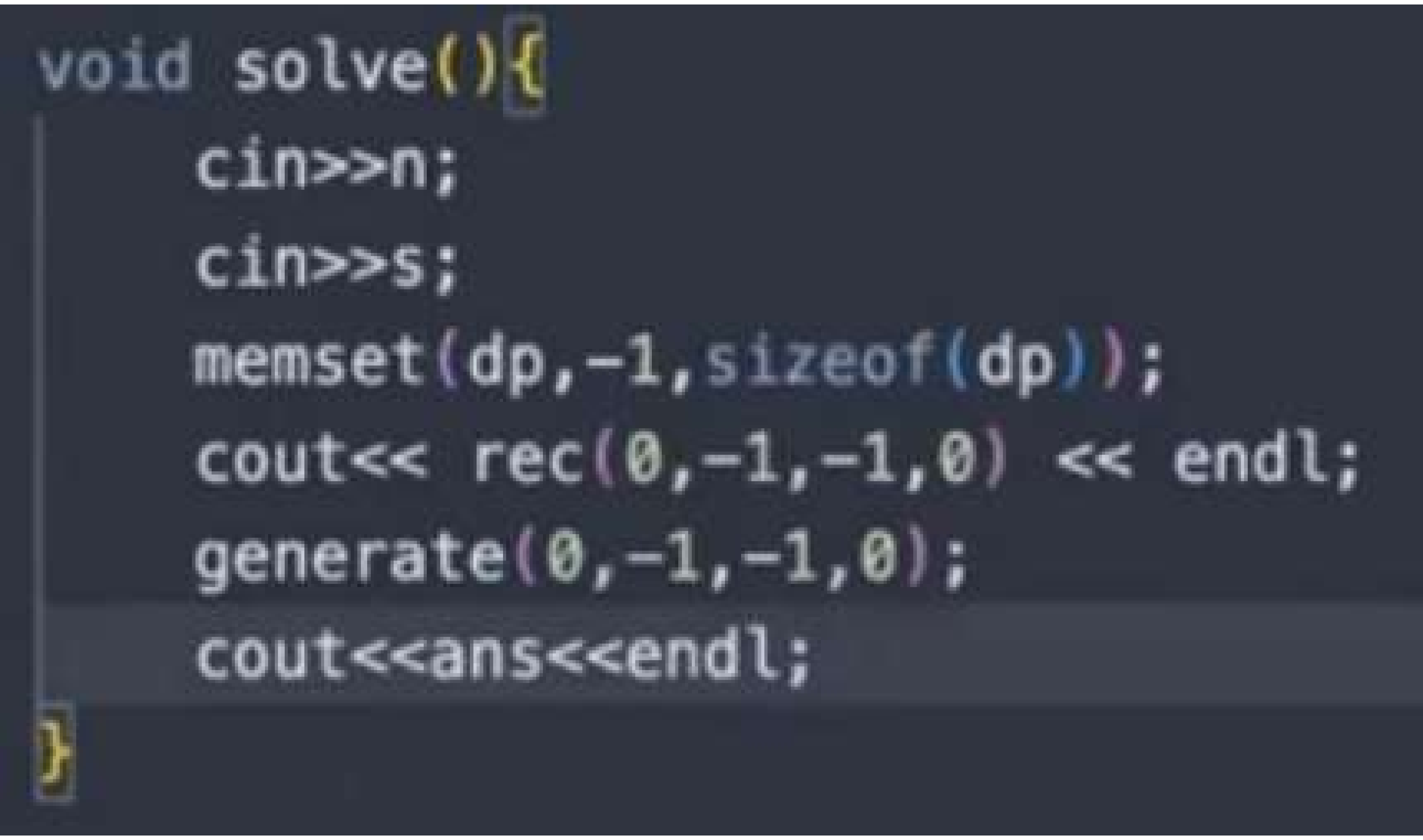
```
// transition
int ans = 0;
// Build choices
set<int> choices;
if(s[i]=='?')choices = {0,1,2,3};
else choices = {s[i]-'A'};

if(prev!=-1)choices.erase(prev);

for(auto v:choices){
    int nfst = fst;
    if(i==0) nfst = v;
    if(v==0){// A case
        ans += rec(i+1, v, nfst, (cnta^1));
    }else{
        ans += rec(i+1, v, nfst, cnta);
    }
}
// save and return
if(prev!=-1) dp[i][prev][fst][cnta] = ans;
return ans;
```



```cpp
string ans;
void generate(int i,int prev,int fst, int cnta){
    // basecase
    if(i==n){
        return;
    }
    // Build choices
    set<int> choices;
    if(s[i]=='?')choices = {0,1,2,3};
    else choices = {s[i]-'A'};

    if(prev!=-1)choices.erase(prev);

    for(auto v:choices){
        int nfst = fst;
        if(i==0) nfst = v;
        if(v==0){// A case
            if(rec(i+1, v, nfst, (cnta^1))>0){
                ans += char('A'+v);
                generate(i+1, v, nfst, (cnta^1));
                return;
            }
```



```
            }
        }else{
            if(rec(i+1, v, nfst, cnta)>0){
                ans += char('A'+v);
                generate(i+1, v, nfst, cnta);
                return;
            }
        }
    }
}
```



```
void solve(){
    cin>>n;
    cin>>s;
    memset(dp,-1,sizeof(dp));
    cout<< rec(0,-1,-1,0) << endl;
    generate(0,-1,-1,0);
    cout<<ans<<endl;
}
```