```
#include <iostream>
#include <math.h>
#include <tuple>
#include <cstdio>
#define N 3
using namespace std;
int A[N][N]= {{6,3,2},{4,20,5},{10,8,15}};
tuple<int,int> mem[N][N];
bool exists = false;
tuple<int,int> answer;
void fillMem(){
        for(int i = 0;i <N;i++){
                for(int j = 0;j<N;j++){
                        mem[i][j] = make_tuple(-1,-1);
                }
        }
}
tuple<int,int> defaultTuple(int neginf,int posinf){
        return make_tuple(neginf,posinf);
}
tuple<int,int> path(int i,int j){
        if(i == (N-1) && (j == N-1)){
                printf("REACHED i=%d,j=%d\n",i,j);
                tuple<int,int> reacho = make_tuple(0,0);
                return reacho;
        }
        printf("Entering i=%d,j=%d.With val: %d\n",i,j,A[i][j]);
        auto memo = mem[i][j];
        if(get<0>(memo) != -1 && get<1>(memo) != -1){
                printf("Remembered for  i=%d,j=%d. ITS : fac3=%d,fac5=%d\n",i,j,get<0>
(memo),get<1>(memo));
                return memo;
        }
        //else if we havent reached we branch
        int fac3up = (A[i+1][j]%3 == 0)? 1 : 0;
        int fac5up = (A[i+1][j]%5 == 0)? 1: 0;

        int fac3right = (A[i][j+1]%3 == 0)? 1: 0;
        int fac5right = (A[i][j+1]%5 == 0)? 1: 0;

        auto bestup = defaultTuple(-999999,999999);
        auto bestright = defaultTuple(-999999,999999);
        if(i+1 < N){
                 bestup = path(i+1,j);
                 get<0>(bestup) = get<0>(bestup) + fac3up;
                 get<1>(bestup) = get<1>(bestup) + fac5up;
        }
        if(j+1 < N){
                 bestright = path(i,j+1);
                 get<0>(bestright) = get<0>(bestright) + fac3right;
                 get<1>(bestright) = get<1>(bestright) + fac5right;
        }
        int diffUp = get<0>(bestup) - get<1>(bestup);
        int diffRight = get<0>(bestright) - get<1>(bestright);

        auto which = (diffUp >= diffRight)?bestup:bestright;
        printf("At i=%d,j=%d val=%d\nWe get fac3=%d,fac5=%d\n",i,j,A[i][j],get<0>
(which),get<1>(which));
        mem[i][j] = which;
        return which;//TODO change that
}
int main(){

        fillMem();
```

```
        int firstFac3 = (A[0][0]%3 == 0) ?1:0;
        int firstFac5 = (A[0][0]%5 == 0) ?1:0;
        auto bestp = path(0,0);
        get<0>(bestp) = get<0>(bestp) + firstFac3;
        get<1>(bestp) = get<1>(bestp) + firstFac5;

        int best3 = get<0>(bestp);
        int best5= get<1>(bestp);
        printf("Our best3=%d,best5=%d\n",best3,best5);
        if(best3>best5){
                printf("Yes there is a monotonic path that visits more 3-entries than
5-entries\n");
                printf("With 3-entries=%d, and 5-entries=%d.\n",best3,best5);
        }


        return 0;
}
```