به نام خداوند بخشنده ی مهربان

راه حل سوال سوم

خب این سوال دو قسمت داره قسمت اول ساختن ۲^۲۵ حالت مختلفی که میتواند اتاق تمساح ها داشته باشد و قسمت دوم اینکه هر حالتی که میسازیم چک کنیم آیا میتوان در آن با شرایط مساله از خانه ی پایین به خانه بالا برسیم یا خیر.

برای قسمت اول مساله ما با استفاده از توابع بازگشتی عمل میکنیم . توابع بازگشتی بسیار مفید میباشند و در بسیاری از سوال های مرحله ۳ کمک میکنند .

قبل اینکه قسمت اول مساله را توضیح دهیم مثال ساده تری میزنیم ، فرض کنید میخواهیم تمام اعداد باینری n رقمی را بسازیم و چاپ کنیم برای حل این سوال یک آرایه تعریف میکنیم که خانه ی اول آن بیانگر رقم اول عدد باینری ما و خانه ی دوم آن بیانگر رقم دوم و ...

حال فرض کنید تابعی داریم که اگر آنرا صدا بزنیم خانه های دوم تا n ام آرایه را تمام حالت هایی که میتواند داشته باشد با صفر یک ها ((2^(n-1)) را بسازد و هر حالتی را که میسازد خانه های یکم تا n ام را چاپ کند . فرض کنید اسم این تابع باشد ;()f اگر ما در ابتدا خانه ی یک را برابر ۰ قرار دهیم و این تابع را صدا بزنیم تمام اعدا باینری n-1رقمی را میسازد و یک صفر اول آن میگذارد و چاپ میکند و بعد از این اگر خانه ی اول را یک بگذاریم و دوباره f را صدا بزنیم تمام اعداد n-1 رقمی را میسازد و یک یک میگذارد اول آن و آنهارا چاپ میکند که در این صورت ما تمام اعداد باینری n رقمی را چاپ کرده ایم.برا ی روشن تر شدن به مثال زیر توجه کنید :

فرض کنید اعداد باینری ۲ رقمی را داشته باشیم :

۰۰

۰۱

۱۰

۱۱

حال اگر یکبار اول آنها صفر و سپس اول آنها یک بگذاریم و آنها را چاپ کنیم تمام اعداد ۳ رقمی باینری ساخته میشود:

۰۰۰

۰۰۱

۰۱۰

۰۱۱

۱۰۰

۱۰۱

۱۱۰

۱۱۱

حال از این ایده استفاده میکنیم تابع f(int k) را تعریف میکنیم و هنگامی که ورودی k را به آن میدهیم فرض میکنیم این تابع تمام حالت های مختلف خانه های k تا n را میسازد و هر حالتی را که ساخت تمام خانه های ۱ تا n را چاپ میکند . اگر توجه کنید این گونه کد زدن بسیار مانند استقرا میماند . فرض میکنیم چیزی درست است چیز کلی تری را با آن مینویسم ، و به تبع مثل هر استقرایی باید یک پایه نیز داشته باشیم و آن این است که اگر k برابر n+1 بود خانه های 1 تا n را چاپ کن.

```cpp
#include<iostream>
using namespace std;

int arr[1000],n;

void f(int k){
    if(k==n+1){
        for(int i=1;i<=n;i++)
            cout<<arr[i];
        cout<<endl;
        return;
    }
    arr[k]=0;
    f(k+1);
    arr[k]=1;
    f(k+1);
}

int main(){
    cin>>n;
    f(1);


    int x;
    cin>>x;
}
```

حال به همین شکل تمام حالت ها را برای این سوال ساخته ایم که به خواندن کد متوجه آن خواهید شد.

اما قسمت دوم سوال ، حال میخواهیم ببینیم هر حالتی که ساخته میشود مطلوب است یا خیر فرض کنید هر حالتی که ساخته میشود یک آرایه دو بعدی است و ۵ در۵ که هر خانه ی آن که برابر صفر باشد یعنی خشکی است و هر خانه ی آن که درون ان یک باشد یعنی درون آن آب است ، حالتی مناسب است که بتوان از خانه ی پایین سمت چپ به خانه ی سمت راست بالا رسید و از دو یک پشت سر هم رد نشویم و همچنین در مسیر حداکثر دو ی ببینیم ، برای حل این سوال میخواهیم برای هر خانه این مقدار را به دست آوریم مسیر با کمترین یک که هیچ دو یک متوالی ای در آن نباشد به آن خانه چند یک دارد.اگر خانه ی بالا سمت راست این مقدارش کمتر از ۳ بود یعنی یک جدول مناسب است . برای اینکه این مقدار را

برای هر خانه به دست آوریم اگر این مقدار برای خانه ی سمت چپش برابر x و این مقدار برای خانه ی سمت راستش y باشد حال مقدار این خانه را میتوان با چند حالت بندی ساده به دست آورد با استفاده از این دو مقدار که در کد این حالت بندی هارا کرده ایم

نکته ی قابل توجه دیگر در کد این است که طوری این مقدار را برای خانه ها پر کرده ایم که هر خانه ای که میخواهد پر شود قبل آن مقدار خانه ی سمت چپ و پایینش حساب شده است.