Soluciones OIFem II Nivel 1

Entrenamiento 7

Formula Recursiva Seis

Teoría

- recusión

Solución

Formamos una función recusiva con las indicaciones, que nos da el enunciado

Link al problema

Código

```cpp
#include <cmath>
#include <iostream>
#include <vector>
using namespace std;
typedef long long int ll;
int f(int n){
  if(n<=20)return 1;
  return f(n-5)+5+n;
}
int main(){
  int n;
  cin>>n;
  cout<<f(n);
  return 0;
}
```

Formula Recursiva Siete

Teoría

- recusión

Solución

Formamos una función recusiva con las indicaciones, que nos da el enunciado

Link al problema

Código

```cpp
#include <iostream>

using namespace std;
int f(int n,int m){
  if(n==5)return f(n-2,m-2)*2;
  if(n<5)return 20;
  return f(n-1,m-1)+2;
}
int main(){
  int n,m;
  cin>>n>>m;
  cout<<f(n,m);
  return 0;
}
```

Formula Recursiva cuarto

Teoría

- recusión
- memorización

Solución

Formamos una función recusiva con las indicaciones, que nos da el enunciado y nos vamos guardando las soluciones con el método de memorización.

Link al problema

Código

```cpp
#include <iostream>
#include <utility>
#include <algorithm>
#include <vector>
#include <cmath>
using namespace std;
int f(int n,vector<int>&sol){
  if(sol[n]!=-1){
    return sol[n];
  }
  if(n<=3)return 1;
  else return sol[n]=f(n-1,sol)+f(n-2,sol)+f(n-3,sol);
}
int main ()
{ int n;
  cin>>n;
  vector<int>sol(n+1,-1);
  cout<<f(n,sol);
  return 0;
}
```

Los frutos del árbol cornalina

Teoría

- recusión

Solución

Tenemos un contador, que inicializamos en 0. En cada paso de la recursión le sumamos 1. Pararemos la recursión cuando el contador sea igual a N, es decir llevemos N pasos de recursión. Además tenemos una variable t, que inicializamos en 1. Cuenta cuantas ramas tenemos hasta el momento. En cada paso de recursión hacemos t*K. Nuestra formula recursiva nos da la cantidad de ramas final. Nuestro output será número de ramas*R.

Link al problema

Código

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <utility>
#include <algorithm>
using namespace std;
typedef long long int ll;
ll ans(int K,int N,int ct,ll t){
        if(ct==N)return t;
        return ans(K,N,ct+1,t*K);
}
int main(){
        ios::sync_with_stdio(false);
        cin.tie(NULL);
        int K,N,R;
        cin>>K>>N>>R;
        cout<<ans(K,N,0,1)*R;
        return 0;
}
```

Reproducción de bichos

Teoría

- recusión

Solución

Haremos una función que calcula sabiendo cuantos bichos hay de cada tipo en un día cuantos bichos habrá al día siguiente (siguiendo las indicaciones del enunciado).Inicializaremos la función con a bichos del tipo A y 0 bichos del tipo B y C. Llamaremos la función d veces de manera recursiva siempre con los nuevos números de bichos.

Link al problema

Código

```cpp
#include <iostream>
#include <utility>
#include <algorithm>
#include <vector>
using namespace std;
int final(int a,int b,int c,int d, int D){
  if(d==D){
    return a+b+c;
  }
  int xa=0,xb=0,xc=0;//nuevos
  xa=a+c;
  xb=3*a+2*b;
  xc=a+4*c+2*b;
  return final(a+xa,b+xb,c+xc,d+1,D);
}
int main ()
{ int A,D;
cin>>A>>D;
cout<<final(A,0,0,0,D);
  return 0;
}
```

Área de la figura

Teoría

- recusión

Solución

Nos damos cuenta de que para formar la figura n, lo que hacemos es coger la figura n-1 y colocamos un cuadrado al lado de cada cuadrado blanco, estos siempre serán (n-1)*4. Es decir:

1.caso base n==1 area=1

2.caso general area(n)=area(n-1)+(n-1)*4

Link al problema

Código

```cpp
#include <cmath>
#include <iostream>
using namespace std;
int poligonointeresante(int n){
  if(n==1)return 1;//caso base
  return poligonointeresante(n-1)+(n-1)*4;//caso general
```

```
}
int main(){
  int n;
  cin>>n;
  cout<<poligonointeresante(n);
  return 0;
}
```

Embaldosando el château

Teoría

- recusión

Solución

La solución será el máximo común divisor de las mededidas a,b dadas. Ya que a será divisble entre mcd y b igual y por tanto se podrá cubrir todo el suelo. Para el mcd hay un algorimo llamado el algoritmo de Euclides, que se puede implementar de manera recursiva. En la página de la OIE está explicado en detalle **Link**. Suele caer en muchos concursos, por lo que es importante entderlo para usarlo rápidamente.

Link al problema

Código

```
#include <iostream>
#include <vector>
#include <cmath>
#include <utility>
#include <algorithm>
using namespace std;
typedef long long int ll;
ll mcd(ll x,ll y){
    if(y==0)return x;
    return mcd(y,x%y);
}
int main(){
    ll n,a,b;
    cin>>n;
    while(n--){
        cin>>a>>b;
        if(b>a)swap(a,b);
        cout<<mcd(a,b)<<"\n";
    }
}
```

Anélidos

Teoría

- recusión
- strings

Solución

Haremos una recusión, que simula las fases de crecimiento. Se llamará la función a si misma hasta llegar al caso base (no hay más fases de crecimiento). En cada pasado por la función crearemos un nuevo string del gusano según las indicaciones, que nos dan en el enunciado del problema.

Link al problema

Código

```cpp
#include <iostream>
#include <vector>
using namespace std;
void crecimiento(string t,int restantes){
  if(restantes==0){// si restantes==0 significa que no nos quedan fases de crecimiento, así que podemos imprimir el gusano
    cout<<t<<"\n";
  }
  else{
  int s=t.size();
  string clon;// aquí guardaremos el gusano depués de la fase de crecimiento actual
  for(int i=0;i<s-1;i++){
    clon.push_back(t[i]);//el anillo actual
    if(t[i]=='A')clon.push_back('N');//anillo contrario
    else clon.push_back('A');
  }
  clon.push_back('C');//la cabeza
  crecimiento(clon,restantes-1);//hemos finalizado una fase de crecimiento, llamamos para la siguiente
  }
}
int main(){
  int n;
  while(cin>>n){
    string t;
    cin>>t;
    if(t[0]=='C')break;
    crecimiento(t,n);
  }
  return 0;
}
```