**Laboratorul 11 – Recursivitate**

/// 3.2. Să se scrie o funcţie recursivă şi una nerecursivă pentru calculul valorii polinoamelor Hermite H(x)

#include <stdio.h>

#include <stdlib.h>

int H(int n, int x){

if(n==0) return 1;

if(n==1) return 2\*x;

return 2\*x\*H(n-1, x)-2\*(n-1)\*H(n-2, x);

}

int main()

{

int n, x, i, h2=1, h1, h;

printf("N = ");

scanf("%d", &n);

printf("X = ");

scanf("%d", &x);

h1=2\*x;

printf("Valoarea functiei este %d", H(n, x));

/// Nerecursiv (ca si in Fibonacci)

for(i=2; i<=n; i++){

h2=h1;

h1=h;

h=2\*x\*h1-2\*(i-1)\*h2;

}

return 0;

}

/// 3.3. Problema turnurilor din Hanoi

#include <stdio.h>

#include <stdlib.h>

void Hanoi(int n, char a, char b, char c){

if(n==1) muta(a, c, n);

else {

Hanoi(n-1, a, c, b);

muta(a, c, n);

Hanoi(n-1, b, a, c);

}

}

void muta(char a, char b, int n){

static int i=1;

printf("%d. %c --> %c (discul #%d)\n", i++, a, b, n);

}

int main()

{

int n;

printf("N = ");

scanf("%d", &n);

Hanoi(n, 'A', 'B', 'C');

return 0;

}

/// 3.4. Să se scrie un program recursiv care citeşte n cuvinte şi le afişează în ordinea inversă a introducerii lor.

#include <stdio.h>

#include <stdlib.h>

void invers(FILE\* pf){

char s[20];

if(fscanf(pf, "%s", s)<1) return ;

invers(pf);

printf("%s ", s);

}

int main()

{

FILE \*pf = fopen("cuvinte.txt", "r");

if(pf==NULL){

perror("cuvinte.txt");

exit(1);

}

invers(pf);

fclose(pf);

return 0;

}

/// 3.5. Să se scrie un program recursiv de generare a produsului cartezian a n mulţimi.

#include <stdio.h>

#include <stdlib.h>

void citire(int a[100], int \*pn, int rez[100]){

int i;

printf("Introduceti n = ");

scanf("%d", pn);

for(i=1; i<=\*pn; i++) rez[i]=0;

for(i=1; i<=\*pn; i++) scanf("%d", &a[i]);

}

void afisare(int \*rez, int n){

for(int i=1; i<=n; i++) printf("%d ", rez[i]);

printf("\n");

}

void back(int a[100], int \*rez, int n, int k){

int i;

for(i=1; i<=a[k]; i++){

rez[k]=i;

if(k==n) afisare(rez, n);

else back(a, rez, n, k+1);

}

}

int main()

{

int n, a[100], rez[100], i;

citire(a, &n, rez);

back(a, rez, n, 1);

return 0;

}

/// 3.6. Să se scrie un program de generare recursivă a submulţimilor de k elemente ale mulţimii A cu n elemente (combinaţiile de n elemente luate câte k).

#include <stdio.h>

#include <stdlib.h>

void citire(int \*pn, int \*pp, int rez[100]){

int i;

printf("Introduceti n = ");

scanf("%d", pn);

printf("Introduceti p = ");

scanf("%d", pp);

for(i=0; i<=\*pn; i++) rez[i]=0;

}

int valid(int \*rez, int k){

int i;

for(i=1; i<=k-1; i++)

if(rez[k]==rez[i]) return 0;

return 1;

}

void afisare(int \*rez, int n){

for(int i=1; i<=n; i++) printf("%d ", rez[i]);

printf("\n");

}

void back(int \*rez, int n, int k, int p){

int i;

if(k==p+1) afisare(rez, p);

else{

if(k>1) rez[k]=rez[k-1];

else rez[k]=0;

while(rez[k]<n-p+k){

rez[k]++;

back(rez, n, k+1, p);

}

}

}

int main()

{

int n, p, rez[100];

citire(&n, &p, rez);

back(rez, n, 1, p);

return 0;

}

/// 3.7. Să se scrie un program de rezolvare a problemei celor 8 regine (determinarea tuturor aşezărilor pe tabla de şah a celor 8 regine astfel încât să nu se atace).

#include <stdio.h>

#include <stdlib.h>

void citire(int \*pn, int rez[100]){

int i;

printf("Introduceti n = ");

scanf("%d", pn);

for(i=0; i<=\*pn; i++) rez[i]=0;

}

int valid(int \*rez, int k){

int i;

for(i=1; i<=k-1; i++)

if(rez[k]==rez[i] || (abs(rez[k]-rez[i])==k-i)) return 0;

return 1;

}

void afisare(int \*rez, int n){

for(int i=1; i<=n; i++) printf("%d ", rez[i]);

printf("\n");

}

void back(int \*rez, int n, int k){

int i;

for(i=1; i<=n; i++){

rez[k]=i;

if(valid(rez, k))

if(k==n) afisare(rez, n);

else back(rez, n, k+1);

}

}

int main()

{

int n, rez[100], i;

citire(&n, rez);

back(rez, n, 1);

return 0;

}

/// 3.8. Să se genereze recursiv permutările mulţimii A de n elemente.

#include <stdio.h>

#include <stdlib.h>

void citire(int \*pn, int rez[100]){

int i;

printf("Introduceti n = ");

scanf("%d", pn);

for(i=0; i<=\*pn; i++) rez[i]=0;

}

int valid(int \*rez, int k){

int i;

for(i=1; i<=k-1; i++)

if(rez[k]==rez[i]) return 0;

return 1;

}

void afisare(int \*rez, int n){

for(int i=1; i<=n; i++) printf("%d ", rez[i]);

printf("\n");

}

void back(int \*rez, int n, int k){

int i;

for(i=1; i<=n; i++){

rez[k]=i;

if(valid(rez, k))

if(k==n) afisare(rez, n);

else back(rez, n, k+1);

}

}

int main()

{

int n, rez[100], i;

citire(&n, rez);

back(rez, n, 1);

return 0;

}

/// 3.9. Se consideră o bară de lungime m şi n repere de lungimi l1, l2, .... , ln. Din bară trebuie tăiate bucăţi de lungimea reperelor date, astfel încât să rezulte din fiecare reper cel puţin o bucată şi pierderea să fie minimă.

#include <stdio.h>

#include <stdlib.h>

int main()

{

return 0;

}

/// 3.11. Să se scrie un program recursiv pentru căutarea eficientă a unei valori într-un tablou care conține numere reale ordonate crescător.

#include <stdio.h>

#include <stdlib.h>

void citire(int a[100], int \*pn, int \*x){

int i;

printf("Introduceti n: ");

scanf("%d", pn);

for(i=0; i<\*pn; i++){

scanf("%d", &a[i]);

}

printf("Introduceti valoarea cautata: ");

scanf("%d", x);

}

int cautareBinara(int \*a, int x, int ls, int ld){

int mij;

if(ls>ld) return -1;

else {

mij=(ls+ld)/2;

if(a[mij]==x) return mij;

if(x<a[mij]) return cautareBinara(a, x, ls, ld-1);

else return cautareBinara(a, x, ls+1, ld);

}

}

int main()

{

int a[100], n, x, rez;

citire(a, &n, &x);

rez=cautareBinara(a, x, 1, n);

if(rez==-1) printf("%d nu a fost gasit in sir", x);

else printf("%d a fost gasit pe pozita %d in sir", x, rez);

return 0;

}

/// 3.12. Să se scrie un program recursiv pentru găsirea eficientă a monedei false dintr-un sac cu 177147 monede. Se știe ca moneda falsă este mai ușoară decât celelalte și că sacul conține doar o astfel de monedă. Aveți la dispoziție doar o balanță cu talere care poate realiza cântăriri precise.

#include <stdio.h>

#include <stdlib.h>

int main(){

/// Divide et impera :)

<https://en.wikipedia.org/wiki/Balance_puzzle>

return 0;

}