Příklad 1/20

Vypočtěte, kolik celkem času zabere jedno zavolání funkce **rekur(4);** za předpokladu, že provedení přikazu **xyz();** trvá vždy jednu milisekundu a že dobu trvání všech ostatních akcí zanedbáme.

```
void rekur(int x) {
  if (x < 1) return;
  rekur(x-1);
  xyz();
  rekur(x-1);
}
```

Příklad 2/20

Určete, jakou hodnotu vypíše program po vykonání příkazu **print(rekur(4));**, když rekurzivní funkce **rekur()** je definována takto:

```
int rekur(int x) {
  if (x < 1) return 2;
  return (rekur(x-1)+rekur(x-1));
}
```

Nedokážete-li výsledek přímo zapsat jako přirozené číslo, stačí jednoduchý výraz pro jeho výpočet.

Příklad 3/20

Funkce

```
int ff(int x, int y) {
 if (x > 0) return ff(x-1,y)+y;
 return 0;
}
```

a) sčítá dvě libovolná celá čísla
b) násobí dvě libovolná celá čísla
c) násobí dvě celá čísla, pokud je první nezáporné
d) vrací nulu za všech okolností
e) vrací nulu nebo y podle toho, zda x je kladné nebo ne

Příklad 4/20

Funkce

```
int ff(int x, int y) {
 if (x > 0) return ff(x-1, y)-1;
 return y;
}
```

a) pro kladná x vrací 0, jinak vrací y
b) odečte x od y, pokud x je nezáporné
c) odečte y od x, pokud x je nezáporné
d) vrací –y pro kladné x, jinak vrací y
e) spočte zbytek po celočíselném dělení y%x

Příklad 5/20

Funkce

```
int ff(int x, int y) {
 if (x < y) return ff(x+1,y);
 return x;
}
```

a) buď hned vrátí první parametr
   nebo jen „do nekonečna“ volá sama sebe
b) vrátí x+1
c) vrátí součet svých parametrů
d) vrátí maximální hodnotu z obou parametrů
e) neprovede ani jednu z předchozích možností

Příklad 6/20

Funkce

```
int ff(int x, int y) {
 if (y>0) return ff(x, y-1)+1;
 return x;
}
```

a) sečte x a y, je-li y nezáporné
b) pro kladná y vrátí y, jinak vrátí x
c) spočte rozdíl x–y, je-li y nezáporné
d) spočte rozdíl y–x, je-li y nezáporné
e) vrátí hodnotu svého většího parametru

Příklad 7/20

```
void ff(int x) {
   if (x > 0) ff(x-1) ;
   abc(x);
   if (x > 0) ff(x-1) ;
}
```

Daná funkce ff je volána s parametrem 2: **ff(2);**. Funkce **abc(x)** je tedy celkem volána:

a) 1 krát
b) 3 krát
c) 5 krát
d) 7 krát
e) 8 krát

Příklad 8/20

Funkce

```
int ff(int x, int y) {
 if (x < y) return ff(x+1,y);
 return x;
}
```

a) buď hned vrátí první parametr
   nebo jen „do nekonečna“ volá sama sebe
b) vrátí maximální hodnotu z obou parametrů
c) vrátí součet svých parametrů
d) vrátí x+1
e) neprovede ani jednu z předchozích možností

Příklad 9/20

Napište rekurzivní funkce, která pro zadané číslo N vypíše řetězec skládající se z N jedniček následovaných 2N dvojkami.

Např. pro N = 3 vypíše 111222222.

Příklad 10/20

Vypište rekurzivně posloupnost čísel:

1 2 3 … N-2 N-1 N N N-1 N-2 … 3 2 1

Příklad 11/20

Sečtěte (odečtěte, vynásobte, vydělte, umocněte) dvě nezáporná celá čísla pomocí rekurzivní (samozřejmě neefektivní) funkce.

Příklad 12/20

Napište rekurzivní funkci, která vypíše pouze hodnoty uložené v listech daného stromu (nebo jen ve vnitřních uzlech).

Příklad 13/20

Posloupnost 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 lze generovat rekurzivní funkcí zavolanou s parametrem 4.

```
void ruler(int val) {
  if (val < 1) return;
  ruler(val-1);
  printf("%d%s",val," ");
  ruler(val-1);
}
```

Zjistěte, co vypíší podobné funkce:

```
void ruler2(int val) {
  if (val < 1) return;
  printf("%d%s",val," ");
  ruler2(val-1);
  ruler2(val-1);
}
```

```
void ruler3(int val) {
  if (val < 1) return;
  ruler3(val-1);
  ruler3(val-1);
  printf("%d%s",val," ");
}
```

Příklad 14/20

Kolik znaků vypíše každá z funkcí v předchozí úloze, spustíme-li ji parameterem 20?

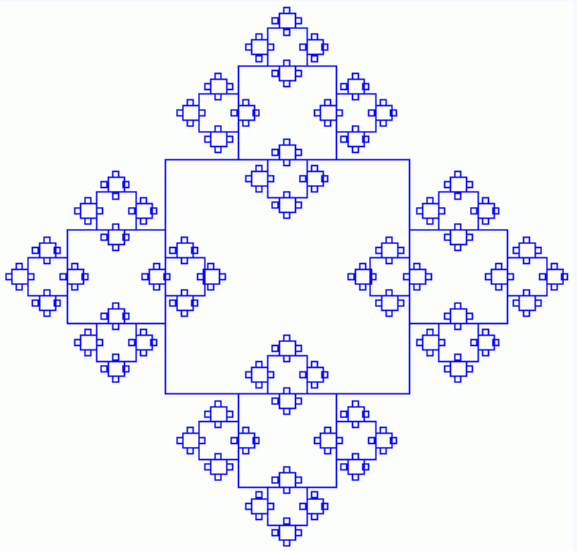
```
void ruler(int val) {
  if (val < 1) return;
  ruler(val-1);
  printf("%d%s",val," ");
  ruler(val-1);
}
```

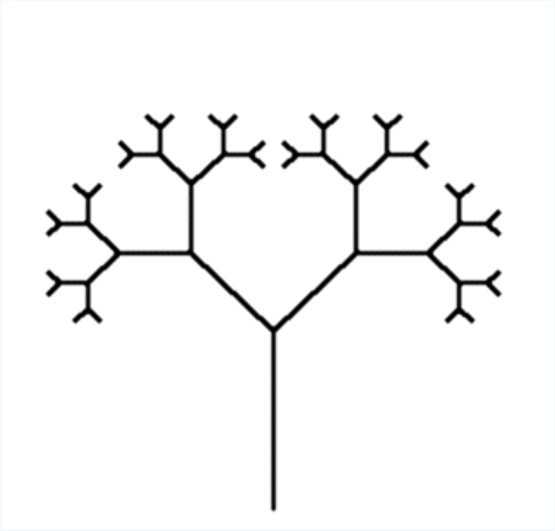
```
void ruler2(int val) {
  if (val < 1) return;
  printf("%d%s",val," ");
  ruler2(val-1);
  ruler2(val-1);
}
```

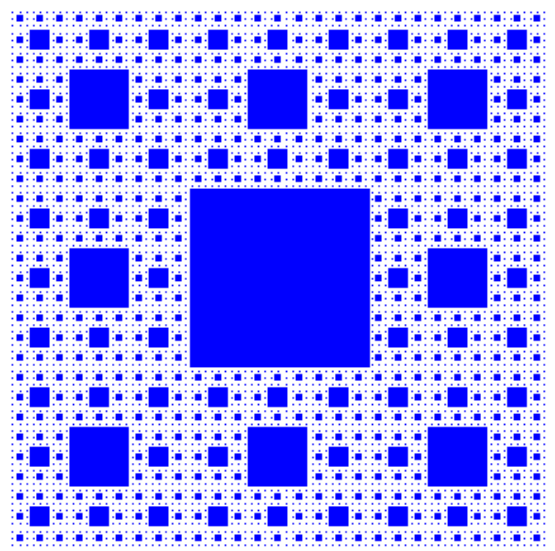
```
void ruler3(int val) {
  if (val < 1) return;
  ruler3(val-1);
  ruler3(val-1);
  printf("%d%s",val," ");
}
```

Příklad 15/20

Formulujte jednoduchý rekurzivní algoritmus, pomocí kterého nakreslíte následující obrázky. Předpokládejte, že příkazy pro kreslení primitiv (úsečky, čtverce, apod.) jsou k dispozici.

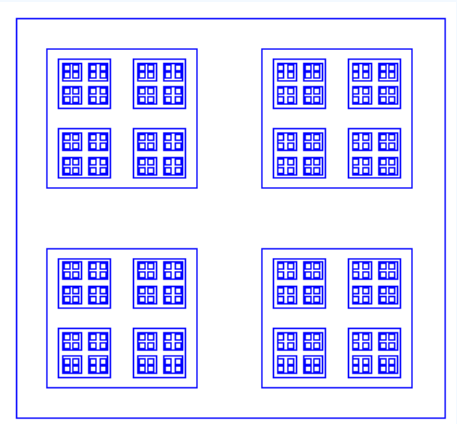


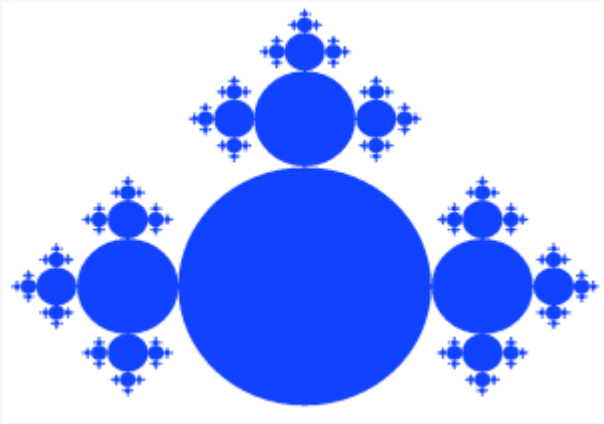


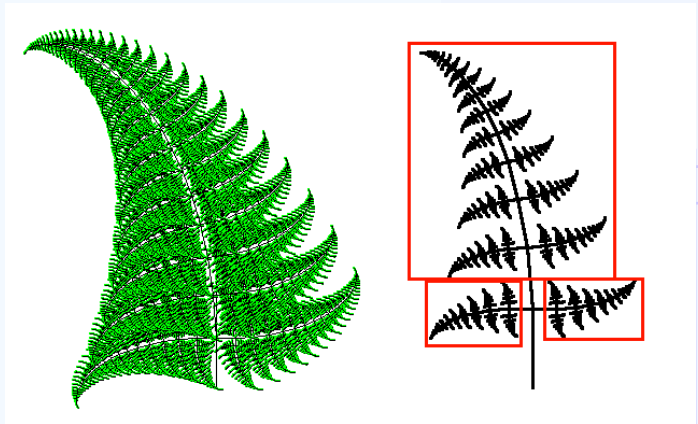


Příklad 15/20

Formulujte jednoduchý rekurzivní algoritmus, pomocí kterého nakreslíte následující obrázky. Předpokládejte, že příkazy pro kreslení primitiv (úsečky, čtverce, apod.) jsou k dispozici.







Příklad 16/20

Sestavte rekurzivní proceduru, která vypíše všechny možnosti rozměnění stokoruny na 1, 2,5,10,20,50 korunová platidla.

Příklad 17/20

Ackermanova A(n, m) funkce je definována níže. Zjistěte, pro která n,m se dá její hodnota vypočítat na běžném počítači (není jich příliš mnoho).

$$A(n, m) = \begin{cases} m+1 & \text{pro } n=0 \\ A(n-1,1) & \text{pro } n>0,\ m=0 \\ A(n-1,A(n,m-1)) & \text{pro } n>0,\ m>0 \end{cases}$$

Příklad 18/20

Schodová posloupnost

Posloupnost celých čísel nazveme schodovou, pokud absloutní hodnota rozdílu každých dvou sousedních prvků je právě 1. Prázdnou posloupnost a posloupnost s jediným prvkem považujeme také za krokové.

Ukázka 1.

a)	1 2 3 4 3 2 1	ANO
b)	1 2 1 2 1 2 3 2 3 4 3 4 5 4 5	ANO
c)	0 –1 –2 –1 –2 –1 0 1 0	ANO
d)	1 2 3 3 4 5 5	NE
e)	8 7 5 4 3 4	NE

Ukázka 2.

Uvažujme nyní jako prvky posloupnosti pouze čísla 0 1 2 a délku posloupnosti rovnou 3. Všech krokových posloupností s těmito parametry je právě 6 a jsou to:

0 1 0, 0 1 2, 1 0 1, 1 2 1, 2 1 0, 2 1 2.

Úloha:

Jsou dána celá čísla 1, 2, 3, ..., N a nezáporné celé číslo L. Napište program, jehož vstupem budou hodnoty N a L a výstupem bude seznam všech schodových posloupností délky L, které obsahují pouze hodnoty 1, 2, 3, ..., N. Použijte rekurzivní funkci.

Příklad 19/20

Daný rekurzivní algoritmus pracuje tak, že pro $n > 1$ data rozdělí na 4 části stejné velikosti, zpracuje 5 těchto částí (tj. jednu z nich dvakrát) a pak jejich řešení spojí. Na samotné rozdělení problému a spojení řešení menších částí potřebuje dobu úměrnou hodnotě $n^2 - n$.

a) Nakreslete první tři úrovně (kořen a dvě další) stromu rekurze.

b) Předpokládejte, že kořen stromu odpovídá činnosti algoritmu nad daty velikost n.

Vypočtěte cenu uzlu v hloubce 2 (=ve 3. úrovni) stromu. Cena uzlu je doba, kterou algoritmus potřebuje na rozdělení dat a sloučení vyřešených podproblémů při velikosti dat, která odpovídá hloubce uzlu.

c) Vypočtěte hloubku stromu rekurze.

Příklad 20/20

Úlohu 19. řešte dále pro případy:

a) Daný rekurzivní algoritmus pracuje tak, že pro $n > 1$ data rozdělí na 3 části stejné velikosti, zpracuje každou tuto část dvakrát a pak jejich řešení spojí. Na samotné rozdělení problému a spojení řešení menších částí potřebuje dobu úměrnou hodnotě $\cdot \log_2(n)$.

b) Daný rekurzivní algoritmus pracuje tak, že pro $n > 1$ data rozdělí na 6 částí stejné velikosti, zpracuje každou tuto část a pak jejich řešení spojí. Na samotné rozdělení problému a spojení řešení menších částí potřebuje dobu úměrnou hodnotě $(n + 1)^2$.

Příklad 20/20

Úlohu 19. řešte dále pro případy:

c) Daný rekurzivní algoritmus pracuje tak, že pro $n > 1$ data rozdělí na 6 částí stejné velikosti, zpracuje 3 tyto části a pak jejich řešení spojí. Na samotné rozdělení problému a spojení řešení menších částí potřebuje dobu úměrnou hodnotě $+ \log_2(n)$.

d) Daný rekurzivní algoritmus pracuje tak, že pro $n > 1$ data rozdělí na 3 části stejné velikosti, zpracuje každou tuto část a pak jejich řešení spojí. Na samotné rozdělení problému a spojení řešení menších částí potřebuje dobu úměrnou hodnotě $(n-1)^2$.