NCP koji prikazuje sve proste brojeve u datom intervalu kojima je zbir cifara složen broj. Interval se zadaje učitavanjem gornje i donje granice (dva prirodna broja). Brojeve prikazati u opadajućem poretku.

```
#include <stdio.h>
#include <stdlib.h>
int prost (int);
int zbirCifara (int);
main() {
  int donja,gornja,i,pom;
  scanf("%d%d", &donja, &gornja);
  if (donja > gornja) {
      pom=donja;
      donja=gornja;
      gornja=pom;  }
  for(i=gornja;i>=donja; i--)
    if (prost (i) && !prost(zbirCifara(i))) printf("%d\n",i);
}
```

NCP koji prikazuje sve proste brojeve u datom intervalu kojima je zbir cifara složen broj. Interval se zadaje učitavanjem gornje i donje granice (dva prirodna broja). Brojeve prikazati u opadajućem poretku.

```
int zbirCifara (int n) {
   int  Suma=0;
   while (n>0)   {
      Suma+= n%10;
       n=n/10; }
   return Suma;
}
```

NCP koji racuna zbir $1 + x + \frac{x^2}{2} + \ldots + \frac{x^n}{n!}$

```
#include<stdio.h>
main() {
   float f, suma, x;
   int i,n;
   scanf("%d", n); scanf("%d", x);
   f = 1;  suma = 1;
   for(i = 1; i <=n; i++) {
      f = f * x / i;
     suma = suma + f; }
  printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

NCP koji racuna zbir $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots - (-1)^n * \frac{x^{2n-1}}{(2n-1)!}$

```
#include<stdio.h>
main() {
  float f, suma, x;
  int i, n;
  scanf("%d", &n); scanf("%f", &x);
  suma = x; f = x;
  for(i = 1; i <= n; i++) {
    f = -f * x * x / ((2*i+1)*2*i);
    suma = suma + f;}
  printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

Šta je rezultat rada sledećeg programa s obzirom na svojstva automatskih i statičkih promenljivih?

```
#include<stdio.h>
void funkcija(void);
main()
{
    int br;
    for(br=1;br<=5;++br) funkcija();
}
void funkcija(void)
{ static int a=0;
  int b=0;
  printf("static =%d auto=%d\n",a,b);
    ++a;
    ++b;
}
```

& bitovsko AND

- 1&1=1   0&1=0   0&0=0
- Neka je b ili 0 ili 1. Onda b & 0 =0, b&1=b
- AKO ZELITE DA NEKE BITOVE POSTAVITE NA NULA, MORATE URADITI OPERACIJU AND SA 0
- Primer: 255 & 15= ?

255 -> 1111 1111

15   -> 0000 1111

255&15   -> 0000 1111

-> oktalno 17

-> heksadekadno 0F

-> dekadno 15

& bitovsko AND

- Primer: Dato je unsigned x; Koliko je x & 01?
- x binarno Bn Bn-1 ....B4 B3 B2 B1 B0

```
x&01= Bn Bn-1  ... B4 B3 B2 B1 B0
    & 0    0    ... 0  0  0  0  1
===================================
      0    0    ... 0  0  0  0  B0
```

- ZAKLJUCAK x & 01 vraća kao rezultat nulti bit broja x

| bitovsko OR

- 1 | 1 = 1, 1 | 0 =1, 0 | 0 = 0
- Ako je b ili 0 ili 1, onda b | 0= b, b | 1 = 1
- AKO ZELITE DA POSTAVITE NEKI BIT NA 1, RADITI OPERACIJU OR SA 1
- Primer: 255 | 15 = ?

255 -> 1111 1111

15 -> 0000 1111

255 | 15 -> 1111 1111

-> oktalno 377

-> heksadekadno FF

-> dekadno 255

& bitovsko AND

- Primer: Dato je unsigned x; Koliko je x | 01?
- x binarno Bn Bn-1 ....B4 B3 B2 B1 B0

```
x|01= Bn Bn-1 ... B4 B3 B2 B1 B0
    | 0    0    ... 0  0  0  0  1
===================================
      Bn Bn-1 ... B4 B3 B2 B1 1
```

- ZAKLJUCAK x | 01 vraca kao rezultat kome je 0-ti bit postavljen na 1

| bitovsko OR

- **Primer: Postaviti 5. bit (B4) na 1, a ostale sacuvati**

```
X Bn Bn-1 ... B6 B5 B4 B3 B2 B1 B0
| 0    0  ... 0  0  1  0  0  0  0  (dekadno 16)
===================================
  Bn Bn-1 ... B6 B5 1 B3 B2 B1 B0
```

- **ZAKLJUCAK: X | 16 postavlja 5. bit u X na 1, a ostale bitove ne menja.**
- **`x | pow(2,n-1)` postavlja n-ti bit u x na 1**
- **Kako postaviti 4. bit u broju x na 0, a ostale ne menjati?**
- **Kako postaviti najniža 3 bita u broju X na nule, a ostale bitove ne dirati?**

^ (caret) bitovsko XOR – ekskluzivna disjunkcija

- 1 ^ 0 = 1, 1 ^ 1 =0, 0 ^ 0 = 0
- Neka je b ili 0 ili 1. Onda onda b ^ 1 = !b , b ^ 0 = b
- Primer: 255 ^ 15= ?

255 -> 1111 1111

15 -> 0000 1111

255^15 -> 1111 0000

-> dekadno 240

^ (caret) bitovsko XOR – ekskluzivna disjunkcija

- **Primer: Dato je unsigned x; Koliko je x ^ 01?**
- **x binarno Bn Bn-1 ....B4 B3 B2 B1 B0**

```
X^01= Bn Bn-1 ... B4 B3 B2 B1 B0
    & 0    0    ... 0  0  0  0  1
===================================
      Bn Bn-1 ... B4 B3 B2 B1 !B0
```

- **ZAKLJUCAK x ^ 01 vraca u kome je 0-ti bit invertovan, a ostali bitovi su nepromenjeni**

~ bitovsko NOT

- Ako je SIZEOF (int) = 1, tj. ako se int registruje u 1 bajtu, tj. u 8 bitova, onda se:
  1 registruje kao 0000 0001
  0 registruje kao 0000 0000
- Tada je ~1 = INVERTOVANO(0000 0001) = 1111 1110 tj. 254
  ~1 = (2 na (sizeof(int)*8)) - 2
- Tada je ~0 = INVERTOVANO (0000 0000) = 1111 1111
  ~0 daje citav registar napunjen bitovima koji su 1

>> SHIFT RIGHT – pomeranje nadesno

- Efekat deljenja stepenom dvojke
- Na primer 32 >> 3 je isto sto i 32 / 2^3,
  ali 32 >> 3 se brže izvršava

<< SHIFT LEFT – pomeranje ulevo

- Efekat množenja stepenom dvojke
- Na primer 3 << 5 je isto što i $3 * 2^5$, ali se 3 << 5 brže izvršava

Šta je rezultat rada sledećeg programa?

```
#include <stdio.h>
main()  {
   printf( "255 & 15 = %d\n", 255 & 15 );
   printf( "255 | 15 = %d\n", 255 | 15 );
   printf( "255 & 15 = %o\n", 255 & 15 );
   printf( "255 | 15 = %x\n", 255 | 15 );
   printf( "255 ^ 15 = %d\n", 255 ^ 15 );
   printf( "4 << 2   = %d\n", 4 << 2 );
   printf( "16 >> 2  = %d\n", 16 >> 2 );
   printf( "~(-3)      = %d\n", ~(-3) );
}
```