II. 3.

```
//declaram alte variabile
char vocale[11]="aeiouAEIOU";

//secventa de instructiuni
//parcurgem sirul de vocale
//daca o vocala nu este in sirul s, o afisam
for (int i=0; i<10; i++)
   if (strchr(s, vocale[i])==NULL)
      cout << vocale[i];
```

III. 1.

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

int factori(int n, int m)
{
   int contor=0;
   int d=2; // incepem cu primul factor prim posibil
   int i, j;

   while (d*d<=n || d*d<=m)
      {
         i=0; j=0; // initializam puterile cu 0

         // determinam puterea lui d in descompunerea lui n
         while (n%d==0)
         {
            i++;
            n/=d;
         }
         // determinam puterea lui d in descompunerea lui m
         while (m%d==0)
         {
            j++;
            m/=d;
         }
         // daca factorul d apare in ambele numere la aceeasi putere strict pozitiva
         if (i>0 && i==j)
            contor++; // creste contorul

      // trecem la urmatorul divizor posibil
      if (d==2)
         d=3;
      else
         d+=2;
   }
   if (n==m && n>1)
      contor++;

   return contor;
}
```

```
int main()
{
   cout<<factori(4, 20)<<endl;
   cout<<factori(16500, 10780)<<endl;
   cout<<factori(2100, 3300)<<endl;
   return 0;
}
```

III. 2.

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

int main() {
   int n;
   int mat[20][20];

   cin>>n;

   // construim matricea
   for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
         mat[i][j]=n-abs(j-(n-1-i));

   // afisam matricea
   for (int i=0; i<n; i++)
   {
      for (int j=0; j<n; j++)
         cout<<mat[i][j]<<" ";
      // trecem la o linie noua
      cout<<endl;
   }

   return 0;
}
```

III.3.

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

int main() {
   ifstream fin("bac.in");

   int nr;
   int pozitie_curenta=0;
   int primul_pozitiv=0;
   int ultimul_pozitiv=0;

   // citim numerele din fisier, pe rand
   while (fin>>nr)
      {
      pozitie_curenta++; // retinem pozitia curenta (pornim de la 1)

      if (nr>0)
      {
         // daca este primul numar pozitiv intalnit, ii salvam pozitia
         if (primul_pozitiv==0)
            primul_pozitiv=pozitie_curenta;

         // actualizam mereu ultima pozitie a unui numar pozitiv
         ultimul_pozitiv=pozitie_curenta;
      }
   }
   fin.close();

   // calculam cele doua lungimi posibile
   int l1=pozitie_curenta-(primul_pozitiv-1);
   int l2=ultimul_pozitiv;

   // determinam lungimea maxima si o afisam
   if (l1>l2)
      cout<<l1<<endl;
   else
      cout<<l2<<endl;

   return 0;
}
```