1. S tipkovnice učitati cijeli broj n iz zatvorenog intervala [1, 10]. Ako broj nije ispravno upisan ispisati poruku *Red matrice je neispravan*. Ako je broj ispravno upisan, ispisati jediničnu matricu reda n. Jedinična matrica reda n se može predstaviti tablicom od n redaka i stupaca u kojima se vrijednost 1 nalazi na glavnoj dijagonali, a vrijednost 0 na svim ostalim mjestima. Pogledati primjere izvršavanja programa.

   Primjeri izvršavanja programa

```
Upisite red matrice > 15↵
Red matrice je neispravan
```

```
5↵
 1 0 0 0 0↵
 0 1 0 0 0↵
 0 0 1 0 0↵
 0 0 0 1 0↵
 0 0 0 0 1↵
```

2. S tipkovnice učitati cijeli broj n iz zatvorenog intervala [1, 10]. Ako broj nije ispravno upisan ispisati poruku *Broj je neispravan*. Ako je broj ispravno upisan, ispisati tablicu od n redaka i stupaca. Koji brojevi se trebaju nalaziti u tablici i prema kojem formatu se ispisuju, zaključiti na temelju primjera izvršavanja programa.

   Primjeri izvršavanja programa

```
Upisite broj > 11↵
Broj je neispravan
```

```
Upisite broj > 1↵
1
```

```
Upisite broj > 5↵
   1   2   3   4   5↵
       6   7   8   9↵
          10  11  12↵
              13  14↵
                  15↵
```

```
Upisite broj > 10↵
   1   2   3   4   5   6   7   8   9  10↵
      11  12  13  14  15  16  17  18  19↵
          20  21  22  23  24  25  26  27↵
              28  29  30  31  32  33  34↵
                  35  36  37  38  39  40↵
                      41  42  43  44  45↵
                          46  47  48  49↵
                              50  51  52↵
                                  53  54↵
                                      55↵
```

3. S tipkovnice učitati cijeli broj m iz zatvorenog intervala [1, 10] i cijeli broj n iz zatvorenog intervala [1, 10]. Nije potrebno provjeravati jesu li brojevi ispravno upisani. Ispisati tablicu od m redaka i n stupaca. Koji brojevi se trebaju nalaziti u tablici i prema kojem formatu se ispisuju, zaključiti na temelju primjera izvršavanja programa.

Primjeri izvršavanja programa

```
Upisite m, n > 5 3↵
   3   2   1↵
   4   3   2↵
   5   4   3↵
   6   5   4↵
   7   6   5↵
```

```
Upisite m, n > 6 7↵
   7   6   5   4   3   2   1↵
   8   7   6   5   4   3   2↵
   9   8   7   6   5   4   3↵
  10   9   8   7   6   5   4↵
  11  10   9   8   7   6   5↵
  12  11  10   9   8   7   6↵
```

```
Upisite m, n > 1 1↵
   1↵
```

4. Ispisati sve Pitagorine trojke čiji su članovi veći od 0 i manji ili jednaki 100. Ispis treba izgledati ovako (objašnjenje: oznaka 3^2 u sljedećem ispisu ima značenje 3^2):

```
1. trojka: 3^2 + 4^2 = 5^2
2. trojka: 4^2 + 3^2 = 5^2
3. trojka: 5^2 + 12^2 = 13^2
   ... itd.
102. trojka: 84^2 + 13^2 = 85^2
103. trojka: 84^2 + 35^2 = 91^2
104. trojka: 96^2 + 28^2 = 100^2
```

Uputa: zadatak se može riješiti tako da se pomoću tri ugniježđene petlje testira svaka kombinaciju 3 cijela broja: 1 1 1; 1 1 2; 1 1 3; ... 1 1 99; 1 1 100; 1 2 1; 1 2 2; ...; 1 2 100; 1 3 1; ... Ispisati samo one kombinacije 3 cijela broja koji zadovoljavaju "uvjet pitagorine trojke".

5. Napisati program koji će ispisati prvih 25 prim brojeva.

   Primjer izvršavanja programa

```
2↵
3↵
5↵
...
79↵
83↵
89↵
97↵
```

6. Napisati program koji će ispisati prim brojeve iz zatvorenog intervala [1000, 10000].

   Primjer izvršavanja programa

```
1009↵
1013↵
1019↵
...
9949↵
9967↵
9973↵
```

7. Algoritam za provjeru je li neki broj prim broj, koji se koristio u dosadašnjim primjerima, može se na jednostavan način znatno unaprijediti:

   - ako cijeli broj n nije djeljiv niti jednim brojem manjim ili jednakim od $n^{1/2}$, tada nije djeljiv niti jednim brojem većim od $n^{1/2}$. To znači da nema potrebe testirati djeljivost s brojevima sve do n-1, dovoljno je testirati djeljivost s brojevima do $n^{1/2}$.
   - ako cijeli broj nije djeljiv s 2, tada nije djeljiv niti jednim parnim brojem većim od 2. To znači da je dovoljno testirati djeljivost s 2, a zatim testirati djeljivost samo neparnim brojevima

   Napisati program kojim će se učitati prirodni broj n. Nije potrebno provjeravati je li upisan ispravan broj. Provjeriti je li broj prim broj testiranjem njegove djeljivosti s 2 i sljedećim neparnim brojevima (dakle 3, 5, 7, ...) manjim ili jednakim $n^{1/2}$.

8. Testirati brzinu izvršavanja originalnog (iz predavanja) programa za testiranje prim broja i poboljšanog programa iz prethodnog zadatka.

   a) programom ispitati je li broj 1000003 prim broj. Uočavate li razliku u brzini originalnog i poboljšanog programa?

   b) programom ispitati je li broj 1645333507 prim broj. Uočavate li razliku u brzini originalnog i poboljšanog programa?

9. S tipkovnice učitavati cijele brojeve dok god se naizmjence upisuje jedan pozitivan, jedan negativan, jedan pozitivan broj, itd. Upisivanje pozitivnog broja nakon pozitivnog ili upisivanje negativnog broja nakon negativnog ili upisivanje nule smatra se pogreškom. U slučaju takve pogreške program ispisuje sumu svih do tada ispravno upisanih brojeva (upisanih prije pogreške) i prekida se njegovo daljnje izvršavanje. Prvi broj koji se upiše s tipkovnice može biti ili pozitivan ili negativan.

Primjeri izvršavanja programa

```
-150↵
23↵
-1↵
130↵
5↵
suma = 2
```

```
-150↵
0↵
suma = -150
```

```
0↵
suma = 0
```

10. Sljedeći program realizirati korištenjem naredbi goto umjesto petlji. **Uz važnu napomenu**: to se radi samo za vježbu jer korištenje naredbe goto nije dopušteno u strukturiranom programiranju.

```
#include <stdio.h>

int main(void) {
   int granica, n, i;

   do {
      printf("Upisite granicu > ");
      scanf("%d", &granica);
      n = granica;
      printf("granica = %d\n", granica);
      while (n > 0) {
         for (i = 1; i <= n; i = i + 1) {
            printf("%4d", i);
         }
         printf("\n");
         n = n - 1;
      }
   } while (granica > 0);

   return 0;
}
```

11. Sljedeći program realizirati bez korištenja naredbi break i continue.

```
#include <stdio.h>
#define GRANICA_1 10
#define GRANICA_2 30

int main(void) {
   int m, n;
   printf("Upisite m, n > ");
   scanf("%d %d", &m, &n);

   do {
      m = m - 1;
      n = n + 1;
      printf("\nm=%d, n=%d\n", m, n);

      if (m < n) {
         printf("prekidam\n");
         break;
      }
      printf("%d + %d = %d\n", m, n, m + n);

      if (m % 2 == 0 || n % 5 == 0) {
         printf("nastavljam\n");
         continue;
      }
      printf("%d * %d = %d\n", m, n, m * n);

   } while (m > GRANICA_1 && n < GRANICA_2);

   return 0;
}
```

12. Čarobni niz (*wondrous numbers*) je beskonačni niz prirodnih brojeva, a_0, a_1, a_2, ..., koji se za prirodni broj *n* definira na sljedeći način:

$$a_i = \begin{cases} n & za\ i = 0 \\ f(a_{i-1}) & za\ i > 0 \end{cases}$$

pri čemu je

$$f(n) = \begin{cases} \frac{n}{2} & ako\ je\ n\ paran \\ 3\,n + 1 & ako\ je\ n\ neparan \end{cases}$$

Drugim riječima, čarobni niz počinje članom $a_0 = n$, a svaki sljedeći član se izračunava na temelju prethodnog, prema sljedećem pravilu: ako je prethodni član bio paran, tada se sljedeći član dobije dijeljenjem prethodnog člana s 2, inače, sljedeći član se dobije množenjem prethodnog člana s 3 i dodavanjem 1.

Primjeri čarobnih nizova brojeva:

za n = 5	5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, ...
za n = 1	1, 4, 2, 1, 4, 2, 1, ...
za n = 12	12, 6, 3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, ...

Napisati program kojim će se s tipkovnice učitati prirodni broj *n* (ne treba provjeravati ispravnost upisanog broja). Zatim na zaslon, po najviše 10 članova u svakom retku, ispisivati članove niza dok se ne ispiše član niza koji ima vrijednost 1. U novi redak ispisati koliko je članova niza ispisano prije pojave člana koji ima vrijednost 1 (broj članova u nizu prije prvog člana niza koji ima vrijednost 1, naziva se *total stopping time*).

Primjeri izvršavanja programa.

```
Upisite prirodni broj n > 7↵
7 22 11 34 17 52 26 13 40 20 ↵
10 5 16 8 4 2 1 ↵
za n = 7, total stopping time = 16↵
```

```
Upisite prirodni broj n > 1↵
1 ↵
za n = 1, total stopping time = 0↵
```

```
Upisite prirodni broj n > 27↵
27 82 41 124 62 31 94 47 142 71 ↵
214 107 322 161 484 242 121 364 182 91 ↵
274 137 412 206 103 310 155 466 233 700 ↵
350 175 526 263 790 395 1186 593 1780 890 ↵
445 1336 668 334 167 502 251 754 377 1132 ↵
566 283 850 425 1276 638 319 958 479 1438 ↵
719 2158 1079 3238 1619 4858 2429 7288 3644 1822 ↵
911 2734 1367 4102 2051 6154 3077 9232 4616 2308 ↵
1154 577 1732 866 433 1300 650 325 976 488 ↵
244 122 61 184 92 46 23 70 35 106 ↵
53 160 80 40 20 10 5 16 8 4 ↵
2 1 ↵
za n = 27, total stopping time = 111↵
```

Rješenja:

1.
```
#include <stdio.h>

int main(void) {
   int n, i, j;

   printf("Upisite red matrice > ");
   scanf("%d", &n);
   if (n < 1 || n > 10) {
      printf("Red matrice je neispravan");
   } else {
      for (i = 0; i < n; i = i + 1) {
         for (j = 0; j < n; j = j + 1) {
            if (i == j) {
               printf("%2d", 1);
            } else {
               printf("%2d", 0);
            }
         }
         printf("\n");
      }
   }
   return 0;
}
```

Uočiti da je, iako "radi", sljedeće rješenje s jednom petljom loše, jer se problem prvo nepotrebno svodi na jednu dimenziju (niz duljine n^2 elemenata), da bi se zatim opet transformirao u dvije dimenzije, pomoću `if (i % n == i / n)` i `if (i > 0 && i % n == n - 1)`.

```
   ...
   } else {
      for (i = 0; i < n * n; i = i + 1) {
         if (i % n == i / n) {
            printf("%2d", 1);
         } else {
            printf("%2d", 0);
         }
         if (i > 0 && i % n == n - 1) {
            printf("\n");
         }
      }
   }
   ...
```

2.
```
#include <stdio.h>

int main(void) {
   int n, i, j, rbr = 1;

   printf("Upisite broj > ");
   scanf("%d", &n);

   if (n < 1 || n > 10) {
      printf("Broj je neispravan");
   } else {
      for (i = 0; i < n; i = i + 1) {
         for (j = 0; j < n; j = j + 1) {
            if (i <= j) {
               printf("%4d", rbr);
               rbr = rbr + 1;
            } else {
               printf("    ", 0);
            }
         }
         printf("\n");
      }
   }

   return 0;
}
```

3.
```
#include <stdio.h>

int main(void) {
   int m, n, i, j;
   printf("Upisite m, n > ");
   scanf("%d %d", &m, &n);

   for (i = 0; i < m; i = i + 1) {
      for (j = n - 1; j >= 0; j = j - 1) {
         printf("%4d", i + 1 + j);
      }
      printf("\n");
   }

   return 0;
}
```

4.
```
#include <stdio.h>

int main(void) {
   int i, j, k, rbr = 0;

   for (i = 1; i <= 100; i = i + 1) {
      for (j = 1; j <= 100; j = j + 1) {
         for (k = 1; k <= 100; k = k + 1) {
            if (i * i + j * j == k * k) {
               rbr = rbr + 1;
               printf("%d. trojka: %d^2 + %d^2 = %d^2\n", rbr, i, j, k);
            }
         }
      }
   }

   return 0;
}
```

5.
```
#include <stdio.h>

int main(void) {
   int n = 2; // broj koji ce se testirati
   int djeljitelj, djeljiv, brojIspisanih = 0;

   while (brojIspisanih < 25) {
      djeljiv = 0;    // hipoteza: n nije djeljiv
      djeljitelj = 2; // pocni testiranje dijeljenjem s 2

      while (djeljitelj <= n - 1 && djeljiv == 0) {
         if (n % djeljitelj == 0) {
            djeljiv = 1;
         }
         djeljitelj = djeljitelj + 1;
      }

      if (djeljiv == 0) {
         printf("%d\n", n);
         brojIspisanih = brojIspisanih + 1;
      }
      n = n + 1;
   }

   return 0;
}
```

6.
```c
#include <stdio.h>
#define DONJA_GR 1000
#define GORNJA_GR 10000

int main(void) {
   int n; // broj koji ce se testirati
   int djeljitelj, djeljiv;

   for (n = DONJA_GR; n <= GORNJA_GR; n = n + 1) {
      djeljiv = 0;    // hipoteza: n nije djeljiv
      djeljitelj = 2; // pocni testiranje dijeljenjem s 2

      while (djeljitelj <= n - 1 && djeljiv == 0) {
         if (n % djeljitelj == 0) {
            djeljiv = 1;
         }
         djeljitelj = djeljitelj + 1;
      }
      if (djeljiv == 0 || n == 1) {
         printf("%d\n", n);
      }
   }

   return 0;
}
```

7.
```c
#include <math.h>
#include <stdio.h>

int main(void) {
   int djeljiv = 0;
   int n, djeljitelj;
   printf("Upisite prirodni broj > ");
   scanf("%d", &n);

   djeljitelj = 2;
   while (djeljitelj <= sqrt(n) && djeljiv == 0) {
      if (n % djeljitelj == 0) {
         djeljiv = 1;
      }
      if (djeljitelj == 2) {
         djeljitelj = djeljitelj + 1;
      } else {
         djeljitelj = djeljitelj + 2;
      }
   }

   if (djeljiv == 1 || n == 1) {
      printf("%d nije prim broj\n", n);
   } else {
      printf("%d jest prim broj\n", n);
   }

   return 0;
}
```

8. a) Pri testiranju broja 1000003 razliku u vremenu izvršavanja je vrlo teško uočiti jer testiranje prim broja u oba programa traje vrlo kratko, nekoliko milisekundi ili manje.

b) Razlika pri testiranju broja 1645333507 je vrlo uočljiva. Originalni program izvršava se približno 4 sekunde, a poboljšani vrlo kratko, približno samo 1 ms (Intel i7-7700 CPU @ 3.60 GHz)

Zainteresirani za ovu temu (dakle, neće se ispitivati na provjerama znanja) u nastavku mogu pročitati na koji se način točnije mjeri vrijeme izvršavanja određenog dijela programa. Na početku i završetku izvršavanja nekog dijela programa u varijable odgovarajućeg tipa zabilježi se vrijednost sistemskog sata (ovog trenutka, bez poznavanja agregatnih tipova podataka, pokazivača i funkcija nije moguće detaljno objasniti koji se tipovi podataka koriste i kako rade funkcije koje bilježe vrijeme). Razlika u zabilježenim vrijednostima se izračuna i ispiše na zaslon.

Sljedeći primjer ilustrira kako se u program mogu dodati naredbe koje omogućuju mjerenje vremena izvršavanja, pri čemu čak nije potrebno razumjeti detalje o tome što se točno dešava. Naredbe koje su radi mjerenja vremena dodane u originalni program, označene su crvenom bojom.

```
#include <math.h>
#include <stdio.h>
#include <sys/timeb.h>

int main(void) {
   int i, n, djeljiv = 0; // hipoteza: nije djeljiv

   printf("Upisite prirodni broj > ");
   scanf("%d", &n);

   // prije pocetka izracunavanja, zabiljezi sistemski sat
   struct timeb pocetak;
   ftime(&pocetak);

   i = 2;
   while (i <= n - 1 && djeljiv == 0) {
      if (n % i == 0) {
         djeljiv = 1; // hipoteza je bila pogresna
      }
      i = i + 1;
   }

   // nakon izracunavanja, ponovo zabiljezi sistemski sat
   struct timeb dovrsetak;
   ftime(&dovrsetak);

   // izracunaj i ispisi razliku zabiljezenih vremena (u milisekundama)
   printf("vrijeme: %lld (ms)\n",
          (long long)(dovrsetak.time * 1000 + dovrsetak.millitm
                    - (pocetak.time * 1000 + pocetak.millitm)));

   if (djeljiv == 1 || n == 1) // jer broj 1 je specijalan slucaj
      printf("%d nije prim broj\n", n);
   else
      printf("%d jest prim broj\n", n);

   return 0;
}
```

9.
```
#include <stdio.h>

int main(void) {
   int predznakPrethKorak = 0; // -1, 1 ovisno o predznaku u prethodnom koraku
   int ispravanUnos;           // 1 ispravan, 0 neispravan unos novog broja
   int suma = 0;
   int novi; // novi ucitani broj

   do {
      scanf("%d", &novi);
      if ((predznakPrethKorak == 0 && novi != 0) ||
          predznakPrethKorak * novi < 0) {
         ispravanUnos = 1;
         suma += novi;
         if (novi > 0) {
            predznakPrethKorak = 1;
         } else if (novi < 0) {
            predznakPrethKorak = -1;
         }
      } else {
         ispravanUnos = 0;
      }
   } while (ispravanUnos == 1);

   printf("suma = %d", suma);

   return 0;
}
```

10.
```
#include <stdio.h>

int main(void) {
   int granica, n, i;

ponovi_do:
   printf("Upisite granicu > ");
   scanf("%d", &granica);
   n = granica;
   printf("granica = %d\n", granica);

ponovi_while:
   if (n > 0) {
      i = 1;

   ponovi_for:
      if (i <= n) {
         printf("%4d", i);
         i = i + 1;
         goto ponovi_for;
      }

      printf("\n");
      n = n - 1;
      goto ponovi_while;
   }

   if (granica > 0) {
      goto ponovi_do;
   }

   return 0;
}
```

11.
```c
#include <stdio.h>
#define GRANICA_1 10
#define GRANICA_2 30

int main(void) {
   int m, n;
   printf("Upisite m, n > ");
   scanf("%d %d", &m, &n);

   do {
      m = m - 1;
      n = n + 1;
      printf("\nm=%d, n=%d\n", m, n);

      if (m < n) {
         printf("prekidam\n");
      } else {
         printf("%d + %d = %d\n", m, n, m + n);

         if (m % 2 == 0 || n % 5 == 0) {
            printf("nastavljam\n");
         } else {
            printf("%d * %d = %d\n", m, n, m * n);
         }
      }

   } while (m > GRANICA_1 && n < GRANICA_2 && m >= n);

   return 0;
}
```

12.
```c
#include <stdio.h>
#define MAX_PO_RETKU 10

int main(void) {
   int n, ai;
   int i = 0;

   printf("Upisite prirodni broj n > ");
   scanf("%d", &n);

   do {
      if (i == 0) {
         // izracunaj clan a0
         ai = n;
      } else {
         // izracunaj sljedeci clan ai na temelju prethodnog clana ai
         if (ai % 2 == 0) {
            ai = ai / 2;
         } else {
            ai = ai * 3 + 1;
         }
         // novi red ako je u tekucem retku vec ispisano MAX_PO_RETKU clanova
         if (i % MAX_PO_RETKU == 0) {
            printf("\n");
         }
      }

      printf("%d ", ai);

      i = i + 1;
   } while (ai != 1);

   printf("\nza n = %d, total stopping time = %d\n", n, i - 1);

   return 0;
}
```