1. Što će se ispisati izvršavanjem sljedećeg programa?

```
#include <stdio.h>

void f(int *p, int n) {
   printf ("%d\n", *(p + n));
}

int main(void) {
   int polje[] = {2, 11, 23, 29, 31, 37};
   int *pp;
   pp = &polje[0];
   f(pp++, 2);
   f(pp, 2);
   f(++pp, 2);

   return 0;
}
```

2. Što će se ispisati izvršavanjem sljedećeg programa?

```
#include <stdio.h>

int *f(int *p) {
   return p - *p;
}

int main(void) {
   int polje[5][3] = {{1, -1, 3},
                      {-3, 2, -2},
                      {0, 5, -4},
                      {-4, -6, 2},
                      {2, 1, 0}
                     };
   int *p = &polje[1][1];
   p = f(p);
   printf ("%d\n", *p);

   p = f(p + *p);
   printf ("%d\n", *p);

   return 0;
}
```

3. Napisati funkciju `zbroj2D` koja kao rezultat vraća zbroj svih članova matrice (članovi matrice su tipa `double`) zadanih dimenzija `m` x `n`. Napisati glavni program (funkciju `main`) tako da s tipkovnice učita dimenzije i članove matrice te na zaslon ispiše rezultat dobiven pozivom funkcije `zbroj2D`, sukladno prikazanom primjeru.

   Primjer izvršavanja programa.

```
Upisite dimenzije > 3 4↵
Upisite clanove >↵
1.2 -4.1 2.2 8.15↵
415.9 1.0 1.0 1.0↵
2.2 2.2 2.2 12.2↵
Suma je: 445.150000
```

4. Napisati funkciju `transpKvad` koja transponira zadanu kvadratnu matricu (pri tome samo mijenja članove zadane matrice, dakle ne stvara novu matricu).

   Napisati glavni program (funkciju `main`) tako da s tipkovnice učita red matrice i članove. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `transpKvad` sukladno prikazanom primjeru (za ispis svakog pojedinog člana matrice koristi se format `"%5d"`).

   Primjer izvršavanja programa.

```
Upisite red matrice > 4↵
Upisite clanove >↵
10 20 30 40↵
11 21 31 41↵
12 22 32 42↵
13 23 33 43↵
   10   11   12   13↵
   20   21   22   23↵
   30   31   32   33↵
   40   41   42   43↵
```

5. Napisati funkciju `genPrim` koja za zadane parametre `granica` i `n` vraća `n` prim brojeva koji su veći ili jednaki parametru `granica`.

   Napisati glavni program (funkciju `main`) tako da s tipkovnice učita vrijednost donje granice i broj prim brojeva koje treba ispisati. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `genPrim` sukladno prikazanom primjeru (za ispis svakog pojedinog prim broja koristi se format `"%7d"`).

   Primjer izvršavanja programa.

```
Upisite donju granicu i broj prim brojeva > 5000 4↵
   5003↵
   5009↵
   5011↵
   5021↵
```

6. Napisati funkciju `sort1D` koja sortira zadano jednodimenzijsko polje cijelih brojeva. Funkcija također kao parametar prima logičku vrijednost `silazno`. Ako je `silazno` istina, tada članove polja treba sortirati od većih prema manjim vrijednostima, inače, treba ih sortirati uzlazno, od manjih prema većim. Napisati glavni program (funkciju `main`) tako da s tipkovnice učita jedan znak (ako je učitan znak 'S', sortiranje će trebati obaviti silazno, inače uzlazno), dimenziju i članove polja. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `sort1D` (dakle sadržaj poredanog polja), sukladno prikazanom primjeru (za ispis svakog pojedinog člana polja koristi se format `"%d·"`).

Primjeri izvršavanja programa.

```
Upisite smjer poretka (S-silazno) > S↵
Upisite dimenziju > 7↵
Upisite clanove > 7 3 12 4 4 3 5↵
12·7·5·4·4·3·3·
```

```
Upisite smjer poretka (S-silazno) > s↵
Upisite dimenziju > 7↵
Upisite clanove > 7 3 12 4 4 3 5↵
3·3·4·4·5·7·12·
```

7. Napisati funkciju `sortRetke2D` koja unutar svakog retka sortira članove zadanog dvodimenzijskog polja cijelih brojeva. Funkcija također kao parametar prima logičku vrijednost `silazno`. Ako je `silazno` istina, tada članove polja unutar svakog retka treba sortirati od većih prema manjim vrijednostima, inače, treba ih sortirati uzlazno, od manjih prema većim. Za sortiranje članova retka treba koristiti funkciju `sort1D` iz prethodnog zadatka.

   Napisati glavni program (funkciju `main`) tako da s tipkovnice učita jedan znak (ako je učitan znak 'S', sortiranje će trebati obaviti silazno, inače uzlazno), zatim učita dimenzije i članove polja. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `sortRetke2D` (dakle sadržaj polja kojem su članovi unutar redaka sortirani) sukladno prikazanom primjeru (za ispis svakog pojedinog člana polja koristi se format `"%5d"`).

Primjeri izvršavanja programa.

```
Upisite smjer poretka (S-silazno) > S↵
Upisite dimenzije > 2 4↵
Upisite clanove >↵
1 2 3 2↵
5 4 1 2↵
    3    2    2    1↵
    5    4    2    1↵
```

```
Upisite smjer poretka (S-silazno) > s↵
Upisite dimenzije > 2 4↵
Upisite clanove >↵
1 2 3 2↵
5 4 1 2↵
    1    2    2    3↵
    1    2    4    5↵
```

8. Napisati funkciju `negativci` koja za zadano jednodimenzijsko polje vraća sve negativne članove tog polja.

   Napisati glavni program (funkciju `main`) tako da s tipkovnice učita dimenziju i članove polja. Nakon toga pomoću rezultata poziva funkcije `negativci` ispisati negativne članove polja sukladno prikazanom primjeru (za ispis svakog pojedinog člana polja koristi se format `"%d·"`).

   Primjeri izvršavanja programa.

```
Upisite broj clanova > 10↵
Upisite clanove > 1 2 3 2 -1 6 -4 -1 -2 3↵
-1·-4·-1·-2·
```

```
Upisite broj clanova > 10↵
Upisite clanove > 1 2 3 2 1 6 4 1 2 3↵
```

9. Napisati funkciju `prviNegativac` koja za zadano jednodimenzijsko polje vraća pokazivač na prvi pronađeni negativni član u polju.

   Napisati glavni program (funkciju `main`) tako da s tipkovnice učita dimenziju i članove polja. Nakon toga ispisati rezultat poziva funkcije `prviNegativac` sukladno prikazanom primjeru.

   Primjeri izvršavanja programa.

```
Upisite broj clanova > 10↵
Upisite clanove > 1 2 3 2 -1 6 -4 -1 -2 3↵
Prvi negativni je -1
```

```
Upisite broj clanova > 10↵
Upisite clanove > 1 2 3 2 1 6 4 1 2 3↵
Nema negativnih
```

10. Napisati funkciju `izbaciNR` koja zadani niz znakova mijenja tako da iz njega izbaci znak novog retka ('\n') ako se radi o posljednjem znaku u nizu (dakle, ako se znak '\n' nalazi neposredno prije terminatora niza). U glavnom programu (funkciji `main`) učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova, pomoću funkcije `izbaciNR` izbaciti iz niza eventualno učitanu oznaku novog retka, ispisati niz i neposredno iza njega znak uskličnik.

    Primjeri izvršavanja programa.

```
Upisite niz > Niz dugacak tocno 20.↵
Niz dugacak tocno 20!
```

```
Upisite niz > Jedan kraci niz.↵
Jedan kraci niz!
```

```
Upisite niz > ↵
!
```

11. Napisati funkciju `izbaciSamoglase` koja zadani niz znakova mijenja tako da iz njega izbaci sve samoglasnike. U glavnom programu (funkciji `main`) učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova, pomoću funkcije `izbaciNR` izbaciti iz niza eventualno učitanu oznaku novog retka, pomoću funkcije `izbaciSamoglase` iz niza izbaciti sve samoglasnike i ispisati tako promijenjeni niz.

    Primjer izvršavanja programa.

```
Upisite niz > Sofoklova Antigona↵
Sfklv ntgn
```

12. Napisati funkciju `traziZadnjiZnak` koja u zadanom nizu pronalazi zadnju pojavu zadanog znaka i vraća pokazivač na taj znak. Ako u zadanom nizu ne postoji zadani znak, funkcija treba vratiti prikladan rezultat na temelju kojeg će se moći prepoznati da zadanog znaka u nizu nema. U glavnom programu (funkciji `main`) učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova, učitati znak, pomoću funkcije `traziZadnjiZnak` pronaći znak i na zaslon ispisati koliko je pronađeni znak udaljen od početka niza (izraženo u broju bajtova) ili ispisati poruku "U nizu nema zadanog znaka".

    Primjeri izvršavanja programa.

```
Upisite niz > Sofoklova Antigona↵
Upisite znak > o↵
15
```

```
Upisite niz > Sofoklova Antigona↵
Upisite znak > B↵
U nizu nema zadanog znaka
```

13. Napisati funkciju `umetniZnak` koja u zadani niz znakova neposredno ispred zadane pozicije ubacuje zadani znak. Pozicija znaka odgovara indeksu znaka. Funkcija vraća logičku vrijednost *istina* ako je pozicija bila ispravno zadana, inače vraća *laž*. U glavnom programu (funkciji `main`) učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova, poziciju ispred koje treba umetnuti znak i znak. Pomoću funkcije `izbaciNR` izbaciti iz niza eventualno učitanu oznaku novog retka, pomoću funkcije `umetniZnak` umetnuti znak ispred zadane pozicije. Ako je znak uspješno umetnut, ispisati novi sadržaj niza, inače, ispisati poruku "Neispravna pozicija".

Primjeri izvršavanja programa.

```
Upisite niz > Sofoklova Antigona↵
Upisite znak > W↵
Upisite poziciju > 2↵
SoWfoklova Antigona↵
```

```
Upisite niz > ↵
Upisite znak > W↵
Upisite poziciju > 0↵
W
```

```
Upisite niz > Sofoklova Antigona↵
Upisite znak > W↵
Upisite poziciju > 19↵
Neispravna pozicija
```

14. Napisati funkciju `traziPrviSamoglas` koja vraća pokazivač na prvi samoglasnik u zadanom nizu. Ako u zadanom nizu ne postoji niti jedan samoglasnik, funkcija treba vratiti prikladan rezultat na temelju kojeg će se moći prepoznati da samoglasnika u nizu nema. U glavnom programu (funkciji `main`) učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova, pomoću funkcije `traziPrviSamoglas` pronaći i ispisati na zaslon samoglasnik ili poruku "U nizu nema samoglasnika".

Primjeri izvršavanja programa.

```
Upisite niz > Sofoklova Antigona↵
o
```

```
Upisite niz > Nvmbr 12th↵
U nizu nema samoglasnika
```

15. Napisati glavni program (funkciji `main`) kojim će se učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova. Uzastopnim pozivanjem funkcije iz prethodnog zadatka, `traziPrviSamoglas`, na zaslon ispisati sve samoglasnike pronađene u nizu. Uputa: funkciju pozivati uzastopno, ali ne tako da samoglasnik traži uvijek od samog početka niza.

Primjeri izvršavanja programa.

```
Upisite niz > Sofoklova Antigona↵
Svi samoglasnici: oooaAioa
```

```
Upisite niz > Nvmbr 12th↵
Svi samoglasnici:
```

16. Napisati funkciju `brojiVelikaMala` koja vraća broj velikih i broj malih slova u zadanom nizu. U glavnom programu (funkciji `main`) učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova, pomoću funkcije izračunati, a zatim na zaslon ispisati dobivene rezultate.

Primjer izvršavanja programa.

```
Upisite niz > Sofoklova Antigona↵
Velikih: 2
Malih: 15
```

17. Napisati funkciju `stvoriObrnutiNiz` koja za zadani niz `niz1` stvara novi niz znakova `niz2` u kojem su znakovi iz niza `niz1` upisani obrnutim redoslijedom. U glavnom programu (funkciji `main`) učitati niz znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova, pomoću funkcije `izbaciNR` izbaciti iz tog niza eventualno učitanu oznaku novog retka, pomoću funkcije `stvoriObrnutiNiz` načiniti novi niz, a zatim ispisati učitani i "obrnuti" niz.

Primjer izvršavanja programa.

```
Upisite niz > Sofoklova Antigona↵
Originalni: Sofoklova Antigona
Obrnuti   : anogitnA avolkofoS
```

18. Definirati tip podatka `tTocka` kojim se opisuje jedna točka u pravokutnom koordinatnom sustavu (x i y su vrijednosti tipa `double`). Napisati funkciju `udaljToc` koja kao parametre prima dva podatka tipa `tTocka` i izračunava udaljenost među tim točkama (rezultat je tipa `double`). Napisati glavni program koji će u varijable `t1` i `t2` tipa `tTocka` učitati koordinate dviju točaka, zatim pomoću funkcije `udaljToc` izračunati njihovu udaljenost, te izračunatu vrijednost ispisati na zaslon.

Primjer izvršavanja programa.

```
Upisite koordinate 1. tocke > 1 2↵
Upisite koordinate 2. tocke > 3 4↵
2.828427
```

19. Jednako kao prethodni zadatak, ali funkcija `udaljToc` kao parametre prima pokazivače na dva podatka tipa `tTocka`. U kojoj od verzija funkcije će biti korišten manji prostor na stogu i zašto?

Rješenja:

1. -

2. -

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

double zbroj2D(double *mat, int m, int n) {
   int i;
   double suma = 0.;
   // moglo je s dvije petlje, ali nije potrebno jer su svi članovi u memoriji jedan iza drugog
   for (i = 0; i < m * n; ++i) {
      suma += *(mat + i);
   }

   return suma;
}

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

   double mat[m][n];

   printf("Upisite clanove >\n");
   for (i = 0; i < m; ++i) {
      for (j = 0; j < n; ++j) {
         scanf("%lf", &mat[i][j]);
      }
   }

   printf("Suma je: %lf", zbroj2D(&mat[0][0], m, n));

   return 0;
}
```

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

void transpKvad(int *mat, int n) {
   int i, j, pomocna;
   for (i = 0; i < n - 1; ++i) {
      for (j = i + 1; j < n; ++j) {
         pomocna = *(mat + n * i + j);
         *(mat + n * i + j) = *(mat + n * j + i);
         *(mat + n * j + i) = pomocna;
      }
   }
   return;
}

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

   int mat[n][n];

   printf("Upisite clanove >\n");
   for (i = 0; i < n; ++i) {
      for (j = 0; j < n; ++j) {
         scanf("%d", &mat[i][j]);
      }
   }

   transpKvad(&mat[0][0], n);

   for (i = 0; i < n; ++i) {
      for (j = 0; j < n; ++j) {
         printf("%5d", mat[i][j]);
      }
      printf("\n");
   }

   return 0;
}
```

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

void genPrim(int granica, int n, int *rez) {
   bool djeljiv;
   int nadjenoBrojeva = 0, kandidat = granica > 1 ? granica : 2, i;
   while (nadjenoBrojeva < n) {
      i = 2;
      djeljiv = 0;
      while (i <= sqrt(kandidat) && djeljiv == 0) {
         if (kandidat % i == 0) {
            djeljiv = 1;
         }
         i = i + 1;
      }
      if (djeljiv == 0 || kandidat == 1) {
         *(rez + nadjenoBrojeva) = kandidat;
         ++nadjenoBrojeva;
      }
      ++kandidat;
   }
   return;
}

int main(void) {
   int dgr, n, i;
   printf("Upisite donju granicu i broj prim brojeva > ");
   scanf("%d %d", &dgr, &n);

   int primBrojevi[n];

   genPrim(dgr, n, &primBrojevi[0]);

   for (i = 0; i < n; ++i) {
      printf("%7d\n", primBrojevi[i]);
   }

   return 0;
}
```

6.
```
#include <stdio.h>
#include <stdbool.h>

void zamijeni(int *x, int *y) {
   int pom;
   pom = *x;
   *x = *y;
   *y = pom;
   return;
}

void sort1D(int *polje, int n, bool silazno) {
   int ind_min_max, i, j;
   for (i = 0; i < n - 1; ++i) {
      ind_min_max = i + 1;
      if (silazno) {
         for (j = i + 2; j < n; ++j) {
            if (*(polje + j) > *(polje + ind_min_max)) ind_min_max = j;
         }
         if (*(polje + ind_min_max) > *(polje + i)) {
            zamijeni(polje + ind_min_max, polje + i);
         }
      } else {
         for (j = i + 2; j < n; ++j) {
            if (*(polje + j) < *(polje + ind_min_max)) ind_min_max = j;
         }
         if (*(polje + ind_min_max) < *(polje + i)) {
            zamijeni(polje + ind_min_max, polje + i);
         }
      }
   }
   return;
}

int main(void) {
   char smjerSorta;
   int n, i;
   printf("Upisite smjer poretka (S-silazno) > ");
   scanf("%c", &smjerSorta);

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

   int polje[n];
   printf("Upisite clanove > ");

   for (i = 0; i < n; ++i) {
      scanf("%d", &polje[i]);
   }

   sort1D(polje, n, smjerSorta == 'S');

   for (i = 0; i < n; ++i) {
      printf("%d ", polje[i]);
   }

   return 0;
}
```

7.
```
#include <stdio.h>
#include <stdbool.h>

void zamijeni(int *x, int *y) {
   int pom;
   pom = *x;
   *x = *y;
   *y = pom;
   return;
}
void sort1D(int *polje, int n, bool silazno) {
   int ind_min_max, i, j;
   for (i = 0; i < n - 1; ++i) {
      ind_min_max = i + 1;
      if (silazno) {
         for (j = i + 2; j < n; ++j) {
            if (*(polje + j) > *(polje + ind_min_max)) ind_min_max = j;
         }
         if (*(polje + ind_min_max) > *(polje + i)) {
            zamijeni(polje + ind_min_max, polje + i);
         }
      } else {
         for (j = i + 2; j < n; ++j) {
            if (*(polje + j) < *(polje + ind_min_max)) ind_min_max = j;
         }
         if (*(polje + ind_min_max) < *(polje + i)) {
            zamijeni(polje + ind_min_max, polje + i);
         }
      }
   }
   return;
}

void sort2D(int *polje, int m, int n, bool silazno) {
   int i;
   for (i = 0; i < m; ++i) {
      sort1D((polje + n * i + 0), n, silazno);
   }
   return;
}

int main(void) {
   char smjerSorta;
   int m, n, i, j;
   printf("Upisite smjer poretka (S-silazno) > ");
   scanf("%c", &smjerSorta);
   printf("Upisite dimenzije > ");
   scanf("%d %d", &m, &n);

   int mat[m][n];
   printf("Upisite clanove >\n");
   for (i = 0; i < m; ++i) {
      for (j = 0; j < n; ++j) {
         scanf("%d", &mat[i][j]);
      }
   }
   sort2D(&mat[0][0], m, n, smjerSorta == 'S');
   for (i = 0; i < m; ++i) {
      for (j = 0; j < n; ++j) {
         printf("%5d", mat[i][j]);
      }
      printf("\n");
   }
   return 0;
}
```

8.
```
#include <stdio.h>

int negativci(int *polje, int n, int *nadjeniNegativci) {
   int nNegativaca = 0, i;
   for (i = 0; i < n; ++i) {
      if (*(polje + i) < 0) {
         *(nadjeniNegativci + nNegativaca) = *(polje + i);
         ++nNegativaca;
      }
   }
   return nNegativaca;
}

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

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

   int polje[n];
   printf("Upisite clanove > ");

   for (i = 0; i < n; ++i) {
      scanf("%d", &polje[i]);
   }

   int nadjeniNegativci[n];
   nNegativaca = negativci(polje, n, nadjeniNegativci);

   for (i = 0; i < nNegativaca; ++i) {
      printf("%d ", nadjeniNegativci[i]);
   }

   return 0;
}
```

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

int *prviNegativac(int *polje, int n) {
   int i;
   for (i = 0; i < n; ++i) {
      if (*(polje + i) < 0) {
         return polje + i;
      }
   }
   return NULL;
}

int main(void) {
   int n, i, *pokNaNegativca;

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

   int polje[n];
   printf("Upisite clanove > ");

   for (i = 0; i < n; ++i) {
      scanf("%d", &polje[i]);
   }

   pokNaNegativca = prviNegativac(polje, n);
   if (pokNaNegativca == NULL) {
      printf("Nema negativnih");
   } else {
      printf("Prvi negativni je %d", *pokNaNegativca);
   }

   return 0;
}
```

10.
```c
#include <stdio.h>
#define MAX_NIZ 20

// funkcija izbacuje \n neposredno prije terminatora
void izbaciNR(char *niz) {
   while (*niz != '\0') {
      if (*niz == '\n' && *(niz + 1) == '\0') {
         *niz = '\0';
      }
      ++niz;
   }
   return;
}

int main(void) {
   char niz[MAX_NIZ + 1];

   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);

   izbaciNR(niz);
   printf("%s!", niz);

   return 0;
}
```

11.
```c
#include <stdio.h>
#define MAX_NIZ 20

// ovdje copy-paste kod za izbaciNR iz 1. zadatka

void izbaciSamoglase(char *niz) {
   int i = 0, potroseno = 0;
   while (*(niz + i) != '\0') {
      if (*(niz + i) != 'a' && *(niz + i) != 'A' &&
          *(niz + i) != 'e' && *(niz + i) != 'E' &&
          *(niz + i) != 'i' && *(niz + i) != 'I' &&
          *(niz + i) != 'o' && *(niz + i) != 'O' &&
          *(niz + i) != 'u' && *(niz + i) != 'U')
         *(niz + potroseno++) = *(niz + i);
      ++i;
   }
   *(niz + potroseno) = '\0';   // <-- OVO JE VAZNO!
   return;
}

int main(void) {
   char niz[MAX_NIZ + 1];
   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);
   izbaciNR(niz);
   izbaciSamoglase(niz);
   printf("%s", niz);
   return 0;
}
```

12.
```
#include <stdio.h>
#define MAX_NIZ 20

char *traziZadnjiZnak(char *niz, char z) {
   char *zadnjaPojava = NULL;
   while (*niz != '\0') {
      if (*niz == z) {
         zadnjaPojava = niz;
      }
      ++niz;
   }
   return zadnjaPojava;
}

int main(void) {
   char niz[MAX_NIZ + 1];
   char z;
   char *pokNaZadnjeg = NULL;

   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);
   printf("Upisite znak > ");
   scanf("%c", &z);

   pokNaZadnjeg = traziZadnjiZnak(niz, z);
   if (pokNaZadnjeg == NULL) {
      printf("U nizu nema zadanog znaka");
   } else {
      printf("%d", pokNaZadnjeg - niz);
   }

   return 0;
}
```

13.
```
#include <stdio.h>
#include <stdbool.h>
#define MAX_NIZ 20

// ovdje copy-paste kod za izbaciNR iz 1. zadatka

bool umetniZnak(char *niz, int poz, char z) {
   int i, duljina = 0;
   bool uspjesno = 0;
   while (*(niz + duljina) != '\0') {
      ++duljina;
   }
   if (poz >= 0 && poz <= duljina) {
      for (i = duljina; i >= poz; --i)
         *(niz + i + 1) = *(niz + i);
      *(niz + poz) = z;
      uspjesno = 1;
   }
   return uspjesno;
}

int main(void) {
   char niz[MAX_NIZ + 1];
   int poz;
   char z;

   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);
   printf("Upisite znak > ");
   scanf("%c", &z);
   printf("Upisite poziciju > ");
   scanf("%d", &poz);

   izbaciNR(niz);
   if (umetniZnak(niz, poz, z)) {
      printf("%s", niz);
   } else {
      printf("Neispravna pozicija");
   }

   return 0;
}
```

14.
```c
#include <stdio.h>
#define MAX_NIZ 20

char *traziPrviSamoglas(char *niz) {
   while (*niz != '\0') {
      if (*niz == 'a' || *niz == 'A' ||
          *niz == 'e' || *niz == 'E' ||
          *niz == 'i' || *niz == 'I' ||
          *niz == 'o' || *niz == 'O' ||
          *niz == 'u' || *niz == 'U') {
         return niz;
      }
      ++niz;
   }
   return NULL;
}

int main(void) {
   char niz[MAX_NIZ + 1];

   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);

   char *pokNaPrviSamoglas = traziPrviSamoglas(niz);
   if (pokNaPrviSamoglas == NULL) {
      printf("U nizu nema samoglasnika");
   } else {
      printf("%c", *pokNaPrviSamoglas);
   }
   return 0;
}
```

15.
```
#include <stdio.h>
#define MAX_NIZ 20

// ovdje copy-paste kod za funkciju traziPrviSamoglas

int main(void) {
   char niz[MAX_NIZ + 1];

   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);

   printf("Svi samoglasnici: ");
   char *traziOdOvogMjestaNadalje = niz;
   char *pokNaPrviSamoglas = NULL;

   do {
      pokNaPrviSamoglas = traziPrviSamoglas(traziOdOvogMjestaNadalje);
      if (pokNaPrviSamoglas != NULL) {
         printf("%c", *pokNaPrviSamoglas);
         traziOdOvogMjestaNadalje = pokNaPrviSamoglas + 1;
      }
   } while (pokNaPrviSamoglas != NULL);

   return 0;
}
```

16.
```
#include <stdio.h>
#define MAX_NIZ 20

void brojiVelikaMala(char *niz, int *brV, int *brM) {
   *brV = *brM = 0;
   while (*niz != '\0') {
      if (*niz >= 'A' && *niz <= 'Z') {
         ++*brV;
      } else if (*niz >= 'a' && *niz <= 'z') {
         ++*brM;
      }
      ++niz;
  }
  return;
}

int main(void) {
   char niz[MAX_NIZ + 1];

   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);

   int velika, mala;
   brojiVelikaMala(niz, &velika, &mala);

   printf("Velikih: %d\n", velika);
   printf("Malih: %d\n", mala);

   return 0;
}
```

17.
```
#include <stdio.h>
#define MAX_NIZ 20

// ovdje copy-paste kod za izbaciNR iz 1. zadatka

void stvoriObrnutiNiz(char *niz, char *obrnutiNiz) {
   int duljina = 0, i = 0;
   while (*(niz + i) != '\0') {
      ++duljina;
      ++i;
   }
   *(obrnutiNiz + duljina) = '\0';
   for (i = 0; i < duljina; ++i) {
      *(obrnutiNiz + duljina - i - 1) = *(niz + i);
   }
   return;
}

int main(void) {
   char niz[MAX_NIZ + 1];
   char obrnutiNiz[MAX_NIZ + 1];
   printf("Upisite niz > ");
   fgets(niz, MAX_NIZ + 1, stdin);
   izbaciNR(niz);

   stvoriObrnutiNiz(niz, obrnutiNiz);
   printf("Originalni: %s\n", niz);
   printf("Obrnuti   : %s", obrnutiNiz);

   return 0;
}
```

18.
```
#include <stdio.h>
#include <math.h>

typedef struct {double x;
                double y;} tTocka;

double udaljToc(tTocka t1, tTocka t2) {
   double udalj;
   udalj = sqrt(pow(t2.x - t1.x, 2.) + pow(t2.y - t1.y, 2.));
   return udalj;
}

int main(void) {
   tTocka t1, t2;
   printf("Upisite koordinate 1. tocke > ");
   scanf("%lf %lf", &t1.x, &t1.y);
   printf("Upisite koordinate 2. tocke > ");
   scanf("%lf %lf", &t2.x, &t2.y);

   printf("%lf", udaljToc(t1, t2));

   return 0;
}
```

19.
```
#include <stdio.h>
#include <math.h>

typedef struct {double x;
                double y;} tTocka;

double udaljToc(tTocka *t1, tTocka *t2) {
   double udalj;
   udalj = sqrt(pow(t2->x - t1->x, 2.) + pow(t2->y - t1->y, 2.));
   return udalj;
}

int main(void) {
   tTocka t1, t2;
   printf("Upisite koordinate 1. tocke > ");
   scanf("%lf %lf", &t1.x, &t1.y);
   printf("Upisite koordinate 2. tocke > ");
   scanf("%lf %lf", &t2.x, &t2.y);

   printf("%lf", udaljToc(&t1, &t2));

   return 0;
}
```