1. Prevajanje programa.

2. Napišite program, ki na standardni izhod izpiše število 100.

3. Nepreveden program, ki predstavlja rešitev naloge iz prejšnjega razdelka, shranite v datoteko `sto.c` in prevedite z zaporedjem ukazov

```
gcc -c sto.c
gcc -o sto sto.o
```

Če izvorni program v datoteki `sto.c` ni vseboval napak, ste po izvršitvi prvega ukaza na disku ustvarili datoteko `sto.o`, v kateri je preveden program, po izvršitvi drugega ukaza pa ste ustvarili datoteko `sto`, v kateri je izvršljiv program.

Ugotovite, koliko bytov vsebujejo datoteke z izvornim, prevedenim in izvršljivim programom.

4. Napišite program, ki na vašem konkretnem računalniku izpiše dolžino datoteke z izvornim, prevedenim in izršljivim programom, če ga prevedete na način, ki je predstavljen v prejšnjem razdelku. (Opozorilo: rešitev je odvisna od računalnika in operacijskega sistema, na katerih prevajate program, in od prevajalnika, s katerim prevajate program.)

5. Uvodne naloge.

6. Napišite program, ki na standardni izhod izpiše “Fakluteta za racunalnistvo in informatiko”.

7. Napišite program, ki na standardni izhod trikrat izpiše “Fakulteta za racunalnistvo in informatiko”, vsakič v svojo vrstico.

8. Napišite program, ki na standardni izhod tisočkrat izpiše “Fakulteta za racunalnistvo in informatiko”, vsakič v svojo vrstico.

9. Napišite program, ki na standardni izhod v nedogled izpisuje “Fakulteta za racunalnistvo in informatiko”, vsakič v svojo vrstico.

10. Napišite program, ki preveri, ali je število 256203161 praštevilo.

11. Napišite program, ki izpiše vsa praštevila med vključno 1 in 1000.

12. Napišite program, ki izpiše vse praštevilske dvojčke med vključno 1 in 1000. Praštevilski dvojček je par praštevil, med katerima je natanko eno sodo število.

13. Napišite program, ki izpiše vsa popolna števila med vključno 1 in 1000. Popolno število je število, ki je enako vsoti vseh svojih deliteljev z izjemo samega sebe.

14. Napišite program, ki iz standardnega vhoda prebere naravno število v desetiškem sistemu in ga nato izpiše. Iz standardnega vhoda lahko bereta samo s funkcijo *getchar*. Nalogo rešite brez uporabe tabel, predpostavite pa lahko, da bo vpisano število manjše ali enako 2^{16}.

15. Napišite program, ki iz standardnega vhoda prebere naravno število v sestnajstiškem sistemu in ga nato izpiše v desetiškem sistemu. V šestnajstiškem sistemu veljajo števke `0...1`, `A...F` in `a...f`. Iz standardnega vhoda lahko berete samo s funkcijo *getchar*. Nalogo rešite brez uporabe tabel, predpostavite pa lahko, da bo vpisano število manjše ali enako 2^{16}.

16. Napišite program, ki na standardni izhod izpiše naravno število v desetiškem sistemu, pri čemer to število najprej shranite v neko spremenljivo tipa **int**. Na standardni izhod lahko pišete zgolj s funkcijo *putchar*. Nalogo rešite brez uporabe tabel, predpostavite pa lahko, da je izbrano število manjše ali enako 2^{16}.

17. Napišite program, ki na standardni izhod izpiše naravno število v šestnajstiškem sistemu, pri čemer to število najprej shranite v neko spremenljivo tipa **int**. Na standardni izhod lahko pišete zgolj s funkcijo *putchar*. Nalogo rešite brez uporabe tabel, predpostavite pa lahko, da je izbrano število manjše ali enako 2^{16}.

18. Kazalci in tabele.

19. Popravi naslednji (nedelujoč) program tako, da bo izpisal število shranjeno v spremenljivki *stevilo* (4), seveda potom uporabe kazalca.

```
#include <stdio.h>
  int main( )
  {
    int stevilo = 4;
    int *kazalec = krneki;
    printf ("%d\n", *kazalec);
    return 0;
  }
```

20. Napišite program, ki v tabelo shrani vrednosti 0, 1, 2, ..., 9, nato vsebino tabele izpišite na standardni izhod.

21. Napišite program, ki spremenljivki tipa int (s poljubnim imenom) priredi vrednost 33, nato dodajte v program kazalec, ki kaže na naslov prejšnje spremenljivke. Končno izpišite vrednost spremenljivke (število 33) z uporabo kazalca.

22. Sledi navodilom:

1. Dodaj v program celoštevilčno spremenljivko (int) imenovano "a" in priredi tej vrednost 111.
2. Dodaj v program kazalec na celoštevilčno spremenljivko (int*) imenovano "b" in priredi tej vrednost, ki je naslov "a" spremenljivke.
3. Dodaj v program kazalec na kazalec na celoštevilčno spremenljivko (int**) imenovano "c" in priredi tej vrednost, ki je naslov "b" spremenljivke.
4. Dodaj v program kazalec na kazalec na kazalec na celoštevilčno spremenljivko (int***) imenovano "d" in priredi tej vrednost, ki je naslov "c" spremenljivke.
5. Izpiši vrednost spremenljivke "a" ekskluzivno z uporabo spremenljivke "d".

23. Sledi navodilom:

1. Sestavi tabelo število, ki vsebuje kvadrate prvih 17 celoštevilčnih števil (1, 4, 9, 16, ...).
2. Sestavi dodatno tablelo kazalcev na cela števila, kjer bo kazalec na indeksu "i" te tabele kazal na vrednost na indeksu "i" prve tabele.
3. Izpiši vrednosti prve tabele brez, da neposredno dostopaš do te (izpiši vrednosti potom kazalcev).

24. Napiši program, ki izpiše vsa praštevila manjša od 100. To naredi z uporabo tabele. Namig: vrednost na nekem indexu tabele bo povedal, če je praštevilo.

25. Upravljanje spomina.

26. Informativni tekst: za pomoč, pri uporabi ukazov *malloc*, *realloc* in *free*, lahko vtipkate v terminal:

```
man malloc
man realloc
man free
```

27. Napiši program, ki izpiše velikosti (v bajtih) osnovnih tipov v C-ju: short, int, long, float, double, long double, char. Namig: uporabi vgrajeno funkcijo C-ja, katero ime se začne s "size" in zakjluči z "of".

28. Preden začneš to nalogo: reši prejšnjo. Napiši program, ki izpiše velikosti (v bajtih) kazalcev na osnovne tipe v C-ju: short*, int*, long*, float*, double*, long double*, char*. Kaj opaziš? Kakšna je razlika s prejšnjo nalogo?

29. Kopiraj, prevedi in zaženi naslednji program. Če misliš, da se bo pravilno izvedel: se motiš. To ne pomeni, da tvoj računalnik nima dovolj spomina (moj ima 8GiB in vseeno program ne deluje). Zakaj?

```
#include <stdio.h>
  int main( )
  {
    char tabela_na_skladu[100 * 1024 * 1024];      /* alocira 100MiB na sklad */
    tabela_na_skladu[1000] = 0;
  }
```

30. Preden začneš to nalogo: reši prejšnjo. Kopiraj, prevedi in zaženi naslednji program. Če misliš, da se NE bo pravilno izvedel, ker se program v prejšnji nalogi ni: se motiš. Zakaj?

```
#include <stdio.h>
  char tabela_na_staticnem_spominu[100 * 1024 * 1024];      /* alocira 100MiB na staticni spomin */
  int main( )
  {
    tabela_na_staticnem_spominu[1000] = 0;
    return 0;
  }
```

31. Preden začneš to nalogo: reši pred prejšnjo. Kopiraj, prevedi in zaženi naslednji program. Če misliš, da se NE bo pravilno izvedel, ker se program v pred prejšnji nalogi ni: se motiš. Zakaj?

```
#include <stdio.h>
#include <stdlib.h>
  int main( )
  {
    char *tabela_na_kopici = (char *) malloc(100 * 1024 * 1024);      /* alocira 100MiB na kopico */
    tabela_na_kopici[1000] = 0;
    return 0;
  }
```

32. Napiši program, ki:

1. Ima funkcijo, ki prejme kot parameter tabelo realnih števil v dvojni natančnosti in dolžino tabele. Nato izračuna in vrne vsoto števil v tabeli (vedno kot realno število v dvojni natančnosti).
2. Iz standardnega vhoda prebere naravno število (**int**), ki bo predstavljalo velikost neke tabele. Imenujno to število n.
3. Alocira tabelo na kopici (!). Tabela bo vsebovala n spremenljivk realnih števil v dvojni natančnosti (Kateri tip je to?). Kaliko bajtov boš alociral?
4. Iz standardnega vhoda preberi n števil tipa double in jih shrani v prejšno tabelo.
5. Izpiši vsoto tabele s pomočjo funkcije v 1.točki.
6. Izpiši vsoto podtabele, ki vsebuje indekse od 0 do $n/2$ s pomočjo funkcije v 1.točki.
7. Izpiši vsoto podtabele, ki vsebuje indekse od $n/2$ do $n-1$ s pomočjo funkcije v 1.točki.
8. Sprosti alocirano tabelo iz kopice (*free*).

33. Napiši funkcijo, ki vrača tako tabelo, da je dostopnih 10 negativnih indeksov in 10 pozitivnih indeksov (index 0 je pozitiven). Tabela naj bo alocirana na kopici. Vsebovaja bo cela števila. Tabela mora biti uporabna kot v spodnjem programu. Opazi negativne indekse!

```
#include <stdio.h>
#include <stdlib.h>
  int *vrniTabelo( );
  int main( )
  {
    int *tabela = vrniTabelo( );
    for (int i = -10; i < 10; ++i) tabela[i] = i;
    for (int i = -10; i < 10; ++i) printf("%d\n", tabela[i]);
    return 0;
  }
```

34. Deklariramo spremenljivko x tipa int, ki ima vrednost 4. Naslov od te spremenljivke shranimo v spremenljivko p. Nato na naslov p shranimo število 6. Izpiši vrednost spremenljivke x. Zakaj je taka?

35. Poženi in oglej si program. Zakaj sta naslova različna?

```
#include <stdio.h>
  int *getPointer(int n)
  {
    return &n;
  }
  int main( )
  {
    int a = 2;
    int *p_neveljaven = getPointer(a);
    int *p_pravilen = &a;
    printf("neveljaven: %p\npravilen: %p\n", p_neveljaven, p_pravilen);
  }
```

36. Napiši program kjer najprej prebereš število števil, ki boš prebral iz standardnega vhoda (največ 100), nato pa posamezna števila. Števila shrani v tabelo in izpiši maksimalno število ter njegov indeks. Namig: uporabi funkcijo scanf.

37. Napiši program kjer najprej s standarnega vhoda prebereš število elementov nato pa posamezna števila. Tabelo nato izpiši v obratnem vrstnem redu.

38. Napiši program, ki bo najprej ustvaril naključno tabelo. Število elementov v tabeli preberi s standarnega vhoda. Elementi tabele pa naj bodo naključna tevila med 0 in maxrand (maxrand tudi prebereš s standarnega vhoda). Program naj tabelo izpiše, nato pa se prešteje, kolikokrat se pojavi katero število. Izpiše naj frekvence samo tistih elementov, ki se v tabeli pojavijo vsaj enkrat. Namig: lahko ustvarite novo tabelo za frekvence števil, kako velika bo? Namig 2: nakljucno spremenljivko tipa **int** lahko generiraš s funkcijo *rand*( ).

39. Sestavi funkcijo, ki bo iz dane tabele npr. (int tabela[10] = 16,3,5,7,9,128,1,91,2,8;) sestavila in vrnila novo tabelo, v kateri bodo samo sodi elementi prvotne tabele. Namig: Najprej preštej, koliko je sodih elementov v tabeli, nato ustvari tabelo in vanjo prepiši vse sode elemente. V funkciji main() nato to tabelo sodih elemntov se izpiši.

40. Napiši funkcijo *swap*(**int** $*a$, **int** $*b$), ki zamenja vrednosti na naslovih, kjer kažeta kazalca. Nato v *main*( ) kliči to funkcijo z naslovama dveh spremenljivk kot argument. Za zaključit izpiši vrednosti teh dveh spremenljivk.

41. Napišite program, ki najprej prebere dolžino zaporedja, nato pa posamezne elemente. Te shranite v tabelo, v kateri nato uredite samo sode elemente(naraščajoče) lihe pustite pri miru. Uporabite funkcijo iz prejšnje naloge. Tabelo nato izpišite. Primer urejanja tabele na tak način:

```
[2, 6, 7, 3, 2, 7, 9, 1, 4, 5, 6] je originalna tabela
[2, 2, 7, 3, 4, 7, 9, 1, 6, 5, 6] je po kriteriju urejena tabela
```

42. Napiši program, ki iz standarnega vhoda bere števila v tabelo dokler ne prebere številke -1. Nato tabelo izpiši. Program mora delovati za (potencialno) poljubno veliko število vhodnih števil. Namig: *realloc*.

43. Napiši program, ki iz standarnega vhoda bere števila v tabelo dokler ne prebere številke -1. Nato izpiši povprečje teh števil. Nato tabelo izpiši. Program mora delovati za (potencialno) poljubno veliko število vhodnih števil. Namig: *realloc*.

44. Nizi.

45. Informativni tekst: nizi so v C-ju tabele znakov. Običajno se nizi po programih podajajo le kot kazalec na začetek tabele znakov. Konec niza predstavlja znak, ki ima vrednost 0 (to ni znak '0'!).

46. Napiši funkcijo, ki vrne niz `"moka00"`. Niz naj bo alociran na kopici (katero funkcijo boš uporabil za to?). Delovanje funkcije preveri tako, da niz izpišes na standardni izhod.

47. Sledi navodilom:
1. Napiši iterativno funkcijo, ki prejme kot parameter niz in vrne dolžino tega. Naprimer: dolžina niza "Flint" je 5.
2. Funkcijo v prejšnji točki napiši še rekurzivno.
3. Primerjaj delovanje funkcij 1. in 2. točke s ugrajeno funkcijo *strlen*( ) v header-ju "string.h".

48. Napiši program, ki prebere niz znakov in v nizu pretvori vse male črke v velike, ostale znake pa pusti pri miru. Spremenjeno tabelo izpiši.

49. Napiši funkcijo **int** *isPalindrome*(**char** ∗*str*), ki preveri, ali je podani niz palindrom.

50. S standardnega vhoda preberi dve besedi, jih združi in podaj funkciji iz prejšnje naloge. Besedi ne bosta daljši od 20-ih znakov. Ali je novonastala beseda palindrom ? Namig: strcat

51. Primerjaj dva enako dolga niza če nista enaka ugotovi na katerih mestih se razlikujeta (izpiši indekse). Niza preberi s standarnega vhoda.

52. Napiši program, ki bere posamezne znake iz standardnega vhoda dokler ne prebere zaporedja znakov 'e', 't' in 'a'. Nato izpise vse predhodno prebrane znake v eni vrstici (brez "eta" zaporedja), pri tem ne izpiše znake, ki predstavljajo števke. Program mora delovati za poljubno število vhodnih znakov, količina teh ni pa v naprej podana. Nasvet: *isdigit* (ctype.h), *realloc*. Primer 1: `VHOD:`

```
j
a
0
n
e
3
5
z
e
t
a
IZHOD:
janez
```

Primer 2: `VHOD:`

```
e
t
4
a
e
t
a
IZHOD:
eta
```

53. Urejanje tabel.

54. Napiši program, ki prebere iz starndardnega vhoda dolžino tabele (recimo n), alocira prostor na kopici za to tabelo, nato pa prebere še n elementov (**int**) in jih shrani v tabelo. Nato implementiraj še funkcijo **void** *selectSort*(**int** $*t$, **int** n), ki uredi elemente tabele po padajočem vrstnem redu, s postopkom "urejanje z izbiranjem". Izpiši vsebino tabele preden jo urediš in po tem, ko si tabelo uredil.

55. Napiši program, ki prebere iz starndardnega vhoda celo število (recimo n), nato naj prebere še zaporedje n celih števil in izpiše 4. največje prebrano število. Če n je manjši od 4 naj program izpiše le `"NAPAKA"`.

56. Dvodimenzionalne tabele.

57. Napiši program, ki s standardnega vhoda prebere kvadratno matriko, nato pa izpiše seštevek elementov matrike na prvi poddiagonali. Vhodni format matrike poljubno določi.

58. Napiši program, kjer za vsak stolpec v matriki poiščeš in izpišeš maksimalen element ter indeks vrstice v kateri se nahaja. Matriko preberi s standarnega vhoda. Najprej dimenzije nato elemente.

59. Napiši program, ki matrično zmnoži dve matriki. S standardnega vhoda preberi najprej dimenziji za prvo matriko, nato posamezna števila, ki jh shrani v tabelo. Enako naredi še za drugo matriko. Vse tabele alociraj s funkcijo *malloc*.

60. Rekurzija.

61. Napiši program, ki izpiše števila od 1 do 40. Uporabi rekurzijo (brez zank)!

62. Napiši program, ki ti sešteje števila od 1 do n. Število n preberi s standarnega vhoda. Uporabi rekurzijo.

63. S standardnega vhoda preberi neko večmestno število. Program naj izpiše kolikokrat se v tem številu pojavi števka 5. Uporabi rekurzijo(brez zank).

64. Napiši program, ki izpiše elemente tabele. Najprej preberi število elementov nato pa še posamezne elemente. Beri s standarnega vhoda v tabelo. Uporabi rekurzijo.

65. Poišči največje število v tabeli. Uporabi rekurzijo. Namig: Katero je največje število v tabeli velikosti 1?

66. Implementiraj funkcijo *faktoriela*(**int** n), ki rekurzivno izračuna vrednost faktoriele za določen parameter n. Matematična definicija funkcije: $f(n) = n * (n-1) * (n-2) * \ldots *2*1$

67. Implementiraj funkcijo, ki prejme kot parameter (vsaj) tabelo števil in njeno dolžino in rekurzivno (!) izračuna aritmetično vsoto te. Pri nalogi drži se še pravila, da števila na lihih indeksih se odštevajo. Primera:

`vhod 1: 2, 1, 3, 5`
`izhod 1: -1` (ker upoštevamo -1 = +2-1+3-5)
`vhod 2: 3, 4, 1, 9, 11, 4, 0, -5, 7, -11`
`izhod 2: 21` (ker upoštevamo 21 = +3-4+1-9+11-4+0-(-5)+7-(-11))

68. Implementiraj funkcijo **int** f(**int** x), ki je definirana tako:

44; če je x deljiv s 7
f(87) - f(5 * x); če je x negativen in ni deljiv s 7
13 * f(x - 1); sicer (x je pozitiven in ni deljiv s 7)

Nekaj testnih preslikav:
f(56) = 44
f(-44) = -16240224
f(131) = 16336892
f(51) = 7436

69. Implementiraj funkcijo $f(x)$, ki prešteje koliko je vsota vseh praštevil manjših ali enakih od števila x. Seveda to naredi rekurzivno!

Nekaj testnih preslikav:
f(33) = 160
f(21) = 77
f(1000) = 76127

70. Napiši rekurzivno funkcijo, ki ti izračuna n-to fiboaccijevo število.

71. Z vhoda preberi niz znakov, ki ni daljši od 100. Napiši rekurzivno funkcijo, ki preveri , če je dani niz palindrom.

72. Napiši rekurzivno funkcijo, ki vrne število v obratnem vrstnem redu. Npr. funkcija dobi stevilo 65387 in vrne stevilo 78356.

73. Napiši rekurzivno funkcijo, ki izračuna dolžino povezanega seznama.

74. Napiši rekurzivno funkcije, ki prešteje kolikokrat se določen element pojavi v seznamu.

75. Napiši rekurzivno funkcijo, ki izpise elemente seznama v obratnem vrstnem redu.

76. Strukture.

77. Primer uporabe struktur. Struktura *ComplexNumber* prdstavlja eno kompleksno število, kjer spremenljivka r predstavlja realni del števila in spremenljivka i imaginarni del števila. Opazi kako v funkcija *print* dostopa do komponent: metodi dostopa sta si enakovredni.

```
#include <stdio.h>
  struct ComplexNumber {
    double r;
    double i;
  };
  void print(struct ComplexNumber *complex)
  {
    printf("%.1lf + i%.1lf", complex->r, (*complex).i);
  }
  int main( )
  {
    struct ComplexNumber number;
    number.r = 1.2;
    number.i = 3.4;
    print(&number);
    printf("\n");
    return 0;
  }
```

78. Uporabi strukturo kompleksnih števil iz zgornje naloge in opravi naslednje naloge:

1. Implementiraj funkcijo **double** *real*(**struct ComplexNumber** $*c$), ki vrne realno komponento kompleksnega števila podanega kot parameter.
2. Implementiraj funkcijo **double** *imaginary*(**struct ComplexNumber** $*c$), ki vrne imaginarno komponento kompleksnega števila podanega kot parameter.
3. Implementiraj funkcijo **struct ComplexNumber** $*$*sum*(**struct ComplexNumber** $*a$, **struct ComplexNumber** $*b$), ki vrne novo strukturo, seveda alocirano na kopici (!), ki vsebuje seštevek dveh kompleksnih števil podanih kot parameter. Seštevek dveh kompleksnih števil zgleda tako: $(a+ib)+(c+id) = (a+c)+i(b+d)$.
4. Implementiraj funkcijo **struct ComplexNumber** $*$*mul*(**struct ComplexNumber** $*a$, **struct ComplexNumber** $*b$), ki vrne novo strukturo, seveda alocirano na kopici (!), ki vsebuje produkt dveh kompleksnih števil podanih kot parameter. Produkt dveh kompleksnih števil zgleda tako: $(a + ib)(c + id) = (ac - bd) + i(bc + ad)$.
5. Izrečunaj: $((2 + i4) + (3 + i1))((2 + i4)(3 + i1))$. Rezultat naj bi bil: $-60 + i80$.

79. Na standardnem vhodu dobimo podatke o študentu v naslednjem vrstnem redu: ime, teža, starost. Definiraj ustrezno strukturo, ki bo hranila vse te podatke. Pri tem uporabi **typedef** za preimenovanje strukture v tip *Student*. Končno izpiši tudi podatke študenta. Zanimivost za optimizacijo: funkcija *scanf* dovoli avtomatično alociranje spomina na kopici pri branju nizov, to lahko izkoristiš z uporabo *scanf*("%ms", &*ptr*), kjer je *ptr* tipa **char** $*$.

80. Napiši program, ki sešteje dve kompleksni števili s pomočjo struktur. Z razliko prejšnjih vaj ti je prepovedana uporaba kazalcev za podajanje strukture kot argument funkcij. Ustvari tip *Complex* z ukazom **typedef**. Pri tem implementiraj funkcijo: *Complex sum*(*Complex a*, *Complex b*) (opazi, da kazalci niso uporabljeni kot v prejšnjih vajah), ki sešteje dve kompleksni števili. Zakaj ta način podajanja struktur kot parameter ni dober? Kje se bo struktura alocirala? Koliko kopij strukture se bo alociralo?

81. Napiši program, ki ti izračuna razliko med dvema časovnima obdobjema. Na standardnem vhodu dobimo podatke: ura1, minute1, sekunde1, ura2, minute2, sekunde2. Definiraj ustrezno strukturo `CAS`, ki bo hranila podatke za eno časovno obdobje. Nato implementiraj funkcijo: void razlikaMedCasovnimiObdobji(struct CAS* t1, struct CAS* t2, struct CAS * razlika), ki izračuna razliko med dvema obdobjema.

82. Napiši program, ki ugotovi če tvoj operacijski sistem uporablja vejice ali pike za ločevanje celoštevilskega in decimalnega dela pri izpisu realnih števil (možni izpisi: 44.4 ali 44,4). To stori z uporabo funkcije *localeconv*( ), ki je definirana v headerju *locale.h*, in vrača kazalec na strukturo **struct** *lconv*. Spremenljivka *decimal_point* v tej strukturi vsebuje niz (”.” ali ”,”), ta niz izpiši.

83. Rekurzivne podatkovne strukture (seznami).

84. Informativni tekst. S spodnjo strukturo se lahko implementira naivno rekurzivno implementacijo seznama v C-ju. Spremenljivka *element* vsebuje vrednost nekega elemenenta v seznamu. Seveda je lahko tip te spremenljivke različen od **int** (v tem primeru imamo seznam elementov tipa **int**). Spremenljivka *next* pa vsebuje naslov naslednjega vozlišča v seznamu, če naslednjega vozlišča ni: potem bo *next* vsebovala vrednost NULL.

```
typedef struct node {
  struct node *next;
  int element;
} Node;
```

85. POZOR: Svetujem, da si najprej napišeš 2 osnovni funkciji za upravljanje s seznami. To sta funkcija za izpis seznama in funkcija za dodajanje elementa v seznam (dodajanje na zacetek je lažje za implementirat). Implementacijo teh dveh funkcij zahteva naslednja naloga. Funkciji pa se uporabljajo v veliki večini nalog tega poglavja.

86. Z uporabo prej definiranega tipa **Node**:

1. Implementiraj funkcijo **Node** **addFirst*(**Node** **list*, **int** *element*), ki že obstoječemu seznamu doda novo vozlišče (*malloc*), ki bo prvo v novem seznamu in bo vsebovalo *element*. Če imamo seznam 4→7→9→1 →NULL in kličemo funkcijo *list* = *addFirst*(*list*, 5), bo seznam zgledal tako: 5→4→7→9→1 →NULL.
2. Implementiraj funkcijo **void** *printList*(**Node** **list*), ki izpiše vsebino seznama.
3. Implementiraj funkcijo **Node** **removeFirst*(**Node** **list*), ki že obstoječemu seznamu izbriše prvo vozlišče (pri tem tudi dealociraj spomin z *free* funkcijo).Če imamo seznam 4→7→9→1 →NULL in kličemo funkcijo *list* = *removeFirst*(*list*), bo seznam zgledal tako: 7→9→1 →NULL.
4. Implementiraj funkcijo **Node** **addLast*(**Node** **list*, **int** *element*), ki že obstoječemu seznamu doda novo vozlišče (*malloc*), ki bo zadnje v novem seznamu in bo vsebovalo *element*. Če imamo seznam 4→7→9→1 →NULL in kličemo funkcijo *list* = *addLast*(*list*, 5), bo seznam zgledal tako: 4→7→9→1→5 →NULL.
3. Implementiraj funkcijo **Node** **removeLast*(**Node** **list*), ki že obstoječemu seznamu izbriše zadnje vozlišče (pri tem tudi dealociraj spomin z *free* funkcijo).Če imamo seznam 4→7→9→1 →NULL in kličemo funkcijo *list* = *removeLast*(*list*), bo seznam zgledal tako: 4→7→9 →NULL.

87. Napiši funkcijo, ki dobi kot parameter seznam števil in izpise vsoto teh. Nalogo lahko trivialno rešiš rekurzivno.

88. Napiši funkcijo, ki dobi kot parameter seznam in izpise koliko elementov vsebuje (tj. število povezanih **struct Node**). Nalogo lahko trivialno rešiš rekurzivno.

89. Napiši funkcijo, ki dobi kot parameter seznam števil in izpise vsoto elementov seznama na sodih *indeksih*. Prvi element v seznamu ima *index* 0. Nalogo reši rekurzivno. Naprimer če imamo seznam 11→5→3→7→1 →NULL je rešitev 11 + 3 + 1 = 15.

90. Napiši funkcijo, ki dobi kot parameter dva seznama in drugega pritakne na konec prvega. Če imamo seznam 4→7→9→1 →NULL in seznam 2→4 →NULL, potem je rezultat 4→7→9→1→2→4 →NULL.

91. Napiši funkcijo, ki prejme seznam kot parameter, in vrne ta seznam obrnjen. Naprimer: če je vhodni seznam 4→7→9→1 →NULL je izhodni seznam 1→9→7→4 →NULL.

92. Napisi funkcijo, ki prejme seznam kot parameter in iz tega zbriše vse elemente na indeksih, ki so primitivna števila. Naprimer v seznamu 9→8→7→6→5→4→3→2→1→0 →NULL, ki ima dolžino 10, so "primitivni" indeksi 2, 3, 5 in 7, torej bi končni seznam zgledal tako: 9→8→5→3→1→0 →NULL.

93. Napiši funkcijo, ki dobi kot parameter dva seznama in vrne 1, če sta seznama enaka, 0 sicer. Seznama sta enaka, če vsebujeta isto število elementov in imata na enakih indeksih enake elemente. Naprimer: seznam 9→8→5 →NULL in 9→8→5 →NULL sta si enaka, seznam 9→8→5 →NULL in seznam 9→8→3 →NULL si nista enaka.

94. Datoteke.

95. Informativni tekst: Za *osnovno* upravljanje z datotekami uporabljamo naslednje funkcije (za več informaciji uporabi ukaz *man* v terminalu):

1. *fopen*( ): Odpre datoteko.
2. *fclose*( ): Zapre datoteko.
3. *fprintf*( ): Piše v datoteko. Uporablja se enako kot *printf*( ).
4. *fscanf*( ): Bere iz datoteke. Uporablja se enako kot *scanf*( ).
3. *fread*( ): Piše v *binarno* datoteko.
4. *fscanf*( ): Bere iz *binarno* datoteke.

96. Primer uporabe: sledi enostaven program, ki bere iz standardnega vhoda (*stdin*) števila in izpiše podvojene vrednosti teh v datoteko `"out.txt"`. Ko ste zapisali nekaj števil in želite poslati programu znak `EOF` (End Of File) pritisnite v terminalu kombinacijo tipk *Ctrl* + *D*.

```
#include <stdio.h>
  int main( )
  {
    FILE *input = stdin;
    FILE *output = fopen("out.txt", "w");
    while (1) {
      int n;
      if (fscanf(input, "%d", &n) ≡ EOF) break;
      fprintf(output, "%d␣*␣2␣=␣%d\n", n, n * 2);
    }
    fclose(output);
    return 0;
  }
```

97. Napiši program, ki v datoteko z imenom "out.txt" zapiše "Cause i always play to win!".

98. Napiši funkcijo **void** *flow*(**FILE** **in*, **FILE** **out*), ki bere binarno iz *in* in piše binarno v *out*. To naj dela, dokler ne zmanjka podatkov za branje (*fread*( ) vrne 0). Namig: uporabil boš en **while**. Funkcijo lahko testiraš tako, da kličeš *flow*(*stdin*, *stdout*).

99. Napiši program, ki iz datoteke z imenom "out.txt" (prejšnje naloge) prebere vso njegovo vsebino in jo izpiše na standardni izhod. Namig: binarno branje in pisanje.

100. Naloge iz kolokvijev.

101. Napišite deklaracijo tipa **node**, ki vsebuje eno celo število, kazalec na tabelo treh celih števil, tabelo petih kazalcev na cela števila in kazalec na samega sebe.

102. Seznam je definiran s strukturo

```
typedef struct _node {
  int value;
  struct _node *next;
} node;
```

103. Napišite iterativno funkcijo **node** **filter*(**int** *max*, **node** **list*) , ki iz seznama odstrani vsa vozlišca, v katerih je shranjena vrednost *value* večja od podane vrednosti *max*, in vrne tako spremenjen seznam.

104. Dvojisko kopico opišemo s strukturo

```
typedef struct _node {
  int value;
  struct _heap *left, *right;
} node;
```

105. Napišite funkcijo **void** *dump_heap*(**FILE** **file*, **node** **heap*) ki na tekstovno datoteko *file* (za katero predpostavite, da je pravilno odprta in pripravljena za pisanje) izpiše vrednosti *value* vseh vozlišč kopice, ki imajo pod seboj več kot 8 drugih vozlišč. Vrstni red izpisa v datoteko ni pomemben. Napišite še, kako bi to funkcijo klicali, da bi dobili izpis na zaslon.

106. Napišite deklaracijo strukture **node**, ki vsebuje eno celo število in tabelo petih kazalcev na samo sebe.

107. Seznam je definiran s strukturo

```
typedef struct _node {
  int value;
  struct _node *next;
} node;
```

108. Kazalec *next* zadnjega elementa seznama ima vrednost NULL. Napišite funkcijo **void** *loopify*(**node** **list*) , ki kazalec *next* zadnjega elementa usmeri na prvi element v seznamu, ki ima vrednost value enako kot zadnji element.

109. Dvojisko kopico opišemo s strukturo

```
typedef struct _heap {
  int value;
  struct _heap *left, *right;
} heap;
```

110. Napišite funkcijo **void** *sums*(**char** **f*, **heap** **h*) , ki za vsako vozlišče kopice h izpiše vsoto vseh vozlišč pod njim (brez vrednosti *value* v tem vozlišču). Vsote naj funkcija izpiše na datoteko z imenom f (če velja $f \equiv \Lambda$, naj izpiše vsote na standardni izhod). Vsote so lahko izpisane v poljubnem vrstnem redu. Kopice h naj funkcija sums ne spreminja.

111. Izpiti.

112. 1.izpit/1.naloga 2015/16 navodila:glej ucilnico 2015/2016

113. 3.izpit/1.naloga 2015/16 navodila:glej ucilnico 2015/2016