Zadaća 3.

Ova zadaća nosi ukupno 4.5 poena, pri čemu prva 4 zadtaka nose po 0.7 poena, peti zadatak nosi 0.8 poena, a šesti 0.4 poena. Svi zadaci se mogu uraditi na osnovu gradiva sa prvih 8 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je ponedjeljak, 20. V 2024. (do kraja dana).

1. U raznim oblastima nauke i tehnike često se dešava da je vrijednost neke funkcije f poznata samo na nekom konačnom skupu tačaka x_i, $i = 1, 2, \ldots, n$ (tj. poznate su samo vrijednosti $y_i = f(x_i)$, $i = 1, 2, \ldots, n$), a zanima nas njena vrijednost $f(x)$ u nekoj tački x koja *ne pripada ovom skupu*. Ovaj problem poznat je kao *problem interpolacije*. Jedan od najjednostavnijih načina za rješavanje ovog problema je da se pretpostavi da je grafik funkcije izlomljena linija koja se dobija spajanjem dužima tačaka (x_i, y_i), $i = 1, 2, \ldots, n$ (tzv. *čvorova interpolacije*) sortiranih u rastući poredak po x koordinati pravim linijama. Na taj način se dobija tzv. *linearna interpolacija*. Međutim, linearna interpolacija ima prilično malu tačnost. Samo neznatno složenija, a neuporedivo tačnija, je tehnika poznata pod nazivom *kubna interpolacija*, kod koje se čvorovi između sebe ne povezuju pravim nego zakrivljenim linijama, koje su lûkovi nekih krivulja opisanih polinomima trećeg stepena (tj. kubnim polinomima, odakle i potiče naziv kubna interpolacija). Mada postoje brojne druge (i komplikovanije) metode interpolacije koje daju neuporedivo bolju tačnost, kubna interpolacija se prilično često koristi zbog svoje jednostavnosti i znatno bolje tačnosti u odnosu na jednostavniju linearnu interpolaciju. Rezultati koji se pritom dobijaju sasvim su prihvatljivi u slučaju kada čvorovi interpolacije nisu previše daleko jedan od drugog i kada možemo tolerirati izvjesne "lomove" grafika funkcije u čvorovima interpolacije.

   Sada ćemo opisati kako se tačno izvodi kubna interpolacija. Ova vrsta interpolacije zahtijeva da čvorovi (x_i, y_i), $i = 1, 2, \ldots, n$ budu sortirani u rastući poredak po x-koordinati, odnosno mora biti ispunjen uvjet $x_1 < x_2 < \ldots < x_n$. Također, moramo imati barem četiri čvora (tj. mora biti $n \geq 4$), a sve vrijednosti x_i, $i = 1, 2, \ldots, n$ moraju biti međusobno različite. Stoga ćemo pretpostaviti da su svi ovi uvjeti ispunjeni. Vrijednost x za koju se vrši procjena mora pripadati intervalu $[x_1, x_n]$ (tj. mora vrijediti $x_1 \leq x \leq x_n$). Sama procjena vrijednosti $f(x)$ se izvodi na sljedeći način. Prvo se pronađe interval $[x_i, x_{i+1})$ koji sadrži tačku x, tj. vrijednost indeksa i takva da je $x_i \leq x < x_{i+1}$. Zatim se vrijednost $f(x)$ procjenjuje koristeći jednačinu polinoma trećeg stepena koji prolazi kroz tačke (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}) i (x_{i+2}, y_{i+2}), tj. kroz dvije tačke koje leže neposredno lijevo i dvije tačke koje leže neposredno desno od razmatrane vrijednosti x. Ne ulazeći u detalje kako se izvodi jednačina takvog polinoma, recimo samo da se vrijednost $f(x)$ procjenjuje koristeći formulu

$$\mathrm{f(x)} \approx \sum_{k=i-1}^{i+2} \left(\mathrm{y}_k \prod_{\substack{\mathrm{j}=i-1 \\ \mathrm{j}\neq k}}^{i+2} \frac{\mathrm{x} - \mathrm{x_j}}{\mathrm{x}_k - \mathrm{x_j}} \right)$$

   Trebamo razmotriti još neke specijalne slučajeve u kojima prethodno opisani opći postupak zakazuje. Prvi specijalan slučaj nastaje kada je $x = x_n$, jer tada odgovarajući interval $[x_i, x_{i+1})$ ne postoji. Međutim, jasno je da u tom specijalnom slučaju imamo $f(x) = y_n$. Drugi specijalni slučaj imamo kada je $i = 1$, jer tada čvor (x_{i-1}, y_{i-1}) ne postoji (to bi trebao biti čvor (x_0, y_0)). Stoga se, u slučaju kada je $i = 1$, radi kao da je $i = 2$ (tj. umjesto dva čvora lijevo i dva čvora desno od razmatrane vrijednosti x uzima se jedan čvor lijevo i tri čvora desno od razmatrane vrijednosti x). Posljednji specijalan slučaj, sličan prethodnom, nastaje kada je $i = n-1$, s obzirom da tada čvor (x_{i+2}, y_{i+2}) ne postoji (to bi trebao biti čvor (x_{n+1}, y_{n+1})). Stoga se, u slučaju kada je $i = n-1$, radi kao da je $i = n-2$ (tj. umjesto dva čvora lijevo i dva čvora desno od razmatrane vrijednosti x uzimaju se tri čvora lijevo i jedan čvor desno od razmatrane vrijednosti x).

   Vaš zadatak je da napravite funkciju za podršku za kubnu interpolaciju. Ova funkcija se treba zvati "`KubnaInterpolacija`", a imaće dvije verzije. Prva verzija ima jedan parametar, koji je vektor uređenih parova (tj. objekata tipa "`std::pair`") realnih brojeva, koji predstavljaju čvorove interpolacije. Kao rezultat, ova funkcija treba vratiti novu funkciju sa jednim realnim argumentom x koja obavlja traženu kubnu interpolaciju za zadanu vrijednost x (to je zapravo funkcija f iz prethodnog teorijskog razmatranja). Na primjer, ukoliko se izvrši naredba

```
auto f = KubnaInterpolacija({{1, 3}, {2, 5}, {4, 4}, {5, 2}, {7, 1}});
```

tada će "f" biti funkcija dobijena kubnom interpolacijom na osnovu čvorova $(1,3)$, $(2,5)$, $(4,4)$, $(5,2)$ i $(7,1)$. Želimo li sada odrediti vrijednost dobijenu interpolacijom u tački $x = 2.5$ (usput, ta vrijednost iznosi $f(2.5) = 5.28125$), možemo izvršiti recimo naredbu

```
std::cout << f(2.5);
```

Kako kubna interpolacija zahtijeva da čvorovi budu sortirani po x-koordinati, ukoliko vektor čvorova ponuđen kao parametar funkciji "KubnaInterpolacija" nije ispravno sortiran, potrebno ga je prvo sortirati u takav poredak, prije nego što nastavimo dalje. U slučaju da među čvorovima interpolacije postoje čvorovi koji imaju identične x-koordinate (što nije dozvoljeno), funkcija "KubnaInterpolacija" treba baciti izuzetak tipa "domain_error" uz prateći tekst "Neispravni cvorovi". Ukoliko vektor čvorova ne sadrži barem četiri čvora, treba baciti isti izuzetak, samo uz prateći tekst "Nedovoljan broj cvorova". Pored toga, funkcija koja se vraća kao rezultat iz funkcije "KubnaInterpolacija" treba da baci izuzetak tipa "range_error" uz prateći tekst "Argument izvan opsega" ukoliko joj se kao argument ponudi vrijednost x takva da je $x < x_1$ ili $x > x_n$.

Druga verzija funkcije "KubnaInterpolacija", s četiri parametra, služi za aproksimaciju neke već postojeće funkcije uz pomoć kubne interpolacije. Prvi parametar ove funkcije je neka realna funkcija jednog realnog argumenta (ili bilo šta što se može pozvati s jednim realnim argumentom i daje rezultat koji se može interpretirati kao realan broj) i on predstavlja funkciju koju želimo da aproksimiramo. Drugi, treći i četvrti parametar ćemo nazvati redom x_{min}, x_{max} i Δx. Ova funkcija treba prvo da kreira čvorove interpolacije na intervalu $[x_{min}, x_{max}]$ sa korakom Δx (tj. u tačkama x_{min}, $x_{min} + \Delta x$, $x_{min} + 2\,\Delta x$, itd.) uzimajući u tim tačkama vrijednosti funkcije zadane prvim parametrom, a zatim treba da iskoristi te čvorove da konstruira novu funkciju dobijenu iz tih čvorova kubnom interpolacijom i da vrati tako konstruisanu funkciju kao rezultat (vodite računa da ukoliko je $x_{max} - x_{min}$ tačno djeljivo sa Δx, tačka x_{max} također treba uključena u čvorove interpolacije, što se možda neće desiti ukoliko budete nepažljivi s realnom aritmetikom). U slučaju da je $x_{min} > x_{max}$ ili da je $\Delta x \leq 0$, treba baciti izuzetak tipa "domain_error" sa pratećim tekstom "Nekorektni parametri".

Napišite i kratki testni program ("main" funkciju) u kojem ćete demonstrirati napisane funkcije, pri čemu ćete drugu funkciju demonstrirati na problemu aproksimacije jedne konkretne funkcije zadane kao $f(x) = x^2 + \sin x + \ln(x+1)$. Slijede primjeri kako trebaju izgledati dijalozi između programa i korisnika (objašnjenja će uslijediti nakon prikaza dijaloga):

```
Odaberite opciju (1 - unos cvorova, 2 - aproksimacija): 1
Unesite broj cvorova: 5
Unesite cvorove kao parove x y: 1 3
2 5
4 4
5 2
7 1
Unesite argument (ili "kraj" za kraj): 1.5
f(1.5) = 4.26042
Unesite argument (ili "kraj" za kraj): 2.5
f(2.5) = 5.28125
Unesite argument (ili "kraj" za kraj): 4.5
f(4.5) = 3
Unesite argument (ili "kraj" za kraj): 0.3
Argument izvan opsega!
Unesite argument (ili "kraj" za kraj): 7
f(7) = 1
Unesite argument (ili "kraj" za kraj): 6
f(6) = 0.6
Unesite argument (ili "kraj" za kraj): kraj
```

```
Odaberite opciju (1 - unos cvorova, 2 - aproksimacija): 2
Unesite krajeve intervala i korak: 0 1 0.2
Unesite argument (ili "kraj" za kraj): 0.9
f(0.9) = 2.23518 fapprox(0.9) = 2.23518
Unesite argument (ili "kraj" za kraj): 1.5
Argument izvan opsega!
Unesite argument (ili "kraj" za kraj): 0.3
f(0.3) = 0.647884 fapprox(0.3) = 0.647955
Unesite argument (ili "kraj" za kraj): kraj
```

Kao što se može vidjeti, nakon nekoliko početnih pitanja, ulazi se u petlju u kojoj se stalno traži vrijednost argumenta x i izračunava vrijednost dobijena interpolacijom za taj argument (u drugom dijalogu prikazuje se i tačna vrijednost funkcije i aproksimativna vrijednost dobijena interpolacijom). U slučaju argumenta izvan opsega, ispisuje se tekst "Argument izvan opsega!". Petlja se prekida čim se unese nešto što nije broj (to ne mora nužno biti riječ "kraj" kako je gore prikazano), a time se ujedno završava i program. Možete pretpostaviti da će pri testiranju na svim drugim mjestima gdje se očekuju brojevi zaista biti uneseni brojevi, tako da ne treba provjeravati validnost unosa. Ukoliko se zadaju takvi početni podaci da funkcija "`KubnaInterpolacija`" baci izuzetak, treba ispisati odgovarajući tekst izuzetka i odmah prekinuti program.

2. U većini programa za obradu teksta postoji opcija za kreiranje *indeksa pojmova*, koji predstavlja popis pojmova koji se nalaze u tekstu, pri čemu se uz svaki pojam navode i pozicije na kojima se taj pojam javlja u tekstu. U ovom zadatku ćete trebati uraditi nešto slično tome. Tekst koji se analizira nalazi se u objektu tipa "`Knjiga`", pri čemu je ovaj tip definiran pomoću deklaracije

```
typedef std::map<std::string, std::vector<std::string>> Knjiga;
```

Dakle, "`Knjiga`" je mapa čija su ključna polja tipa "`string`", a pridružene vrijednosti vektori stringova. Ključna polja predstavljaju oznake poglavlja knjige (oni su stringovnog tipa, jer oznake poglavlja ne moraju nužno biti brojčane). Odgovarajuće pridružene vrijednosti su vektori stringova, pri čemu svaki string u vektoru predstavlja sadržaj jedne stranice razmatranog poglavlja knjige. Na primjer, neka je "`k`" neki objekat tipa "`Knjiga`". Tada je "`k["XIV"][5]`" sadržaj šeste stranice XIV-og poglavlja knjige (šeste a ne pete, jer indeksacija počinje od nule).

Prvo ćete napraviti funkciju "`KreirajIndeksPojmova`" koja kao parametar prima neki objekat tipa "`Knjiga`", koji predstavlja tekst koji se analizira. Funkcija kao rezultat treba vratiti mapu koja predstavlja traženi indeks pojmova. Ključna polja ove mape su tipa "`string`", a pridružene vrijednosti su skupovi čiji su elementi uređene trojke (tipa "`tuple`"), čija je prva koordinata tipa "`string`", a druga i treća su cijeli brojevi. Vrijednosti ključnih polja predstavljaju različite riječi pronađene u analiziranom tekstu, dok su odgovarajuće pridružene vrijednosti skupovi čiji elementi opisuju pozicije na kojima se odgovarajuća riječ nalazi unutar razmatranog teksta. Svaka pozicija opisana je kao uređena trojka, pri čemu prva koordinata predstavlja oznaku poglavlja, druga koordinata redni broj stranice (uz numeraciju koja počinje od jedinice, a ne od nule), dok treća koordinata predstavlja poziciju razmatrane riječi unutar stranice (tj. indeks od kojeg počinje riječ). Na primjer, pretpostavimo radi jednostavnosti da tekst ima samo jedno poglavlje koje se zove "I", u kojem ima samo jedna stranica (koja naravno ima indeks 0), čiji je sadržaj string "ABC QWE STSDA ABC ABC DHI QWE HRKW DHI". Funkcija tada treba da vrati mapu u kojoj se nalazi 5 parova (toliko ukupno ima različitih riječi u tekstu), čija su ključna polja stringovi "ABC", "DHI", "HRKW", "QWE" i "STSDA" (odnosno riječi koje se nalaze u tekstu), dok su odgovarajuće pridružene vrijednosti skupovi {("I", 1, 1), ("I", 1, 15), ("I", 1, 19)}, {("I", 1, 23), ("I", 1, 36)}, {("I", 1, 31)}, {("I", 1, 5), ("I", 1, 27)} i {("I", 1, 9)}. Jasno je zbog čega su u svim trojkama prve dvije koordinate "I" odnosno 1, a što se tiče treće koordinate, riječ "ABC" se nalazi na pozicijama 1, 15 i 19 u razmatranom stringu (pozicije se također broje od jedinice, odnosno prvi znak je na poziciji 1), riječ "DHI" nalazi se na pozicijama 22 i 35, itd.

Sljedeća funkcija koju treba napraviti je "`PretraziIndeksPojmova`". Ova funkcija kao parametar prima neku riječ (tipa "`string`") i mapu koja predstavlja indeks pojmova, a koja kao rezultat vraća odgovarajući skup pozicija za datu riječ pronađen u datom indeksu pojmova, ili prazan skup u slučaju da data riječ nije nađena u indeksu pojmova. U slučaju da parametar ne predstavlja riječ, odnosno da sadrži bilo kakve druge znakove osim slova i cifri, funkcija treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Neispravna rijec".

Konačno, posljednja funkcija koju treba napraviti je "`IspisiIndeksPojmova`". Ova funkcija kao parametar prima mapu koja predstavlja indeks pojmova, a ispisuje njen kompletan sadržaj na ekranu u obliku tako da se u svakom redu ispisuje prvo pojam, zatim dvotačka praćena razmakom, i na kraju, spisak pozicija međusobno razdvojenih zarezom iza kojeg slijedi razmak. Svaka pozicija ispisuje se kao oznaka poglavlja, broj stranice i pozicije unutar stranice, pri čemu se između tih podataka nalazi kosa crta (bez razmaka ispred i iza nje). Na primjer, za indeks pojmova kreiran na osnovu teksta iz prethodnog primjera, ova funkcija bi trebala proizvesti ispis poput sljedećeg:

```
ABC: I/1/1, I/1/15, I/1/19
DHI: I/1/23, I/1/36
HRKW: I/1/31
QWE: I/1/5, I/1/27
STSDA: I/1/9
```

Stringovi koji predstavljaju sadržaje stranica mogu sadržavati ma kakve znakove (slova, cifre i znake interpunkcije), pri čemu se ne pravi razlika između malih i velikih slova, tako da "ABC", "Abc", "abc", "aBc" i "aBC" predstavljaju istu riječ. Pri tome se u mapi uvijek bilježi riječ koja se sastoji samo od velikih slova (tako da će sve ove riječi zapravo biti evidentirane kao riječ "ABC"). Kao riječ unutar teksta smatra se svaka neprekinuta sekvenca znakova koji su slova ili cifre koja je sa obe strane omeđena razmakom ili znakom interpunkcije (tačnije, znakom koji nije niti slovo niti cifra), osim eventualno na početku ili kraju stringa, kad ne mora biti razmaka ili znaka interpunkcije ispred odnosno iza riječi. Na primjer, u stringu "pqr, ab/123 (qwe) tt2 " riječi su "pqr", "ab", "123", "qwe" i "tt2".

Napisane funkcije demonstrirajte u glavnom programu u kojem se prvo sa tastature unosi tekst za analizu (na način koji će biti vidljiv iz primjera koji slijedi), nakon čega se na ekranu ispisuje kreirani indeks pojmova. Potom program ulazi u petlju u kojoj se za svaku riječ unesenu sa tastature ispisuje pozicije na kojima se riječ nalazi u analiziranom tekstu, pri čemu su pozicije međusobno razdvojene razmacima (koristeći pri tome kreirani indeks pojmova), ili informaciju da unesena riječ nije nađena (u vidu teksta "Unesena rijec nije nadjena!"). U slučaju da se unese nešto što nije riječ, program treba da ispiše "Neispravna rijec!". Program treba da prekine rad kada korisnik ne unese ništa, odnosno odmah pritisne ENTER kada se očekuje unos. Na isti način se signalizira i da ne želimo unositi novo poglavlje ili novu stranicu. Dijalog između korisnika i programa trebao bi izgledati poput sljedećeg:

```
Unesite naziv poglavlja: I (ENTER)
Unesite sadržaj stranice 1: abc qwe stsda abc abc dhi qwe hrkw dhi (ENTER)
Unesite sadržaj stranice 2: (ENTER)
Unesite naziv poglavlja: (ENTER)

Kreirani indeks pojmova:
ABC: I/1/1, I/1/15, I/1/19
DHI: I/1/23, I/1/36
HRKW: I/1/31
QWE: I/1/5, I/1/27
STSDA: I/1/9

Unesite rijec: Abc (ENTER)
Rijec nadjena na pozicijama: I/1/1 I/1/15 I/1/19
Unesite rijec: hrkw (ENTER)
Rijec nadjena na pozicijama: I/1/31
Unesite rijec: xyzzy (ENTER)
Rijec nije nadjena!
Unesite rijec: #?@@! (ENTER)
Neispravna rijec!
Unesite rijec: (ENTER)
Dovidjenja!
```

3. Dopunite program za rad sa generičkom strukturom "Matrica" obrađen na Predavanju 8_b sa novom funkcijom "ProsiriPremaFunkcijama". Ova funkcija kao prvi parametar prima matricu čiji su elementi cjelobrojni (tačnije objekat tipa "Matrica<int>" gdje je "Matrica" razvijena generička struktura), kao drugi parametar prima mapu čiji su ključevi tipa "Smjer" koji je pobrojani tip s ograničenim vidokrugom, a pridružene vrijednosti funkcije (tačnije objekti tipa "function") koje primaju jedan cjelobrojni argument (tipa "int") i vraćaju cjelobrojni rezultat, dok kao treći parametar prma cijeli broj n (tipa "int"). Mapa smije imati samo 3 vrijednosti ključnog polja: "Desno", "Dolje" i "Dijagonalno" (ovo su pobrojane konstante tipa "Smjer"), tako da, u suštini, mapa smije sadržavati najviše 3 para. Funkcija treba da vrati kao rezultat novu matricu dobijenu na sljedeći način. Ukoliko se u mapi nalazi par sa ključem "Desno", matrica treba da se proširi zdesna novim elementima koji se dobijaju primjenom funkcije pridružene ključu "Desno" nad elementima izvorne matrice (tako da će matrica postati dvostruko šira). Slično, ukoliko se u mapi nalazi par sa ključem "Dolje", matricu treba proširiti s donje strane novim elementima koji se dobijaju primjenom funkcije pridružene ključu "Dolje". Ukoliko se zahtijeva i proširivanje desno i

proširivanje dolje, odgovarajuću "rupu" koja nastaje dijagonalno dolje desno treba popuniti kopijom izvornih elemenata, osim ukoliko je zadan i par sa ključem "`Dijagonalno`". U tom slučaju elemente u toj "rupi" treba popuniti rezultatima primjene funkcije pridružene ključu "`Dijagonalno`". Isto tako, ukoliko je u mapi prisutan par sa ključem "`Dijagonalno`", a nedostaje makar jedan od parova sa ključevima "`Desno`" ili "`Dolje`", popunjavamo ono što možemo popuniti u skladu sa opisanim postupkom, dok "rupe" koje nastaju popunjavamo kopijom izvornih elemenata. Ukoliko je u mapi prisutan samo par sa ključem "`Desno`" ili samo par sa ključem "`Dolje`", proširivanje matrice vršimo samo na jednu stranu (zdesna odnosno odozdo). Konačno, ukoliko je mapa prazna, ne vršimo nikakvo proširivanje.

Opisani postupak treba ponoviti $n-1$ puta, pri čemu se u svakoj i-toj iteraciji proširuje matrica dobijena u $(i-1)$-oj iteraciji. Pri tome treba naglasiti da je zbog *edukativnih* a ne praktičnih razloga, ovaj postupak potrebno obavljati *tačno ovako kao što je opisano*, odnosno u svakoj iteraciji je potrebno kreirati *novu matricu* i popunjavati je na osnovu matrice dobijene u prethodnoj iteraciji. Drugim riječima, nije dozvoljeno odmah na početku kreirati matricu konačne veličine i popunjavati je postupno na osnovu popunjenog jednog njenog dijela. Razlog za to je što se problem za koji se od studenata očekuje da se izbore s njim neće pojaviti ukoliko se ne bude radilo tačno onako kako je opisano. Na primjer, neka je "`A`" matrica formata 2×2 s elementima

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

i neka je mapa "`m`" deklarirana na sljedeći način:

```
std::map<Smjer, std::function<int(int)>> m{
  {Smjer::Desno, [] (int x) { return x + 1; }},
  {Smjer::Dolje, [] (int x) { return x + 2; }},
  {Smjer::Dijagonalno, [] (int x) { return x + 3; }}
};
```

Nakon poziva funkcije "`ProsiriPremaFunkcijama(A, m, 3)`", matrica koja se dobije kao rezultat treba imati sljedeću strukturu (ali je pri tome prethodno trebala biti kreirana i matrica formata 4×4, čiji su elementi prikazani podebljano u dolje prikazanoj matrici):

$$\begin{pmatrix} \mathbf{1} & \mathbf{2} & \mathbf{2} & \mathbf{3} & 2 & 3 & 3 & 4 \\ \mathbf{3} & \mathbf{4} & \mathbf{4} & \mathbf{5} & 4 & 5 & 5 & 6 \\ \mathbf{3} & \mathbf{4} & \mathbf{4} & \mathbf{5} & 4 & 5 & 5 & 6 \\ \mathbf{5} & \mathbf{6} & \mathbf{6} & \mathbf{7} & 6 & 7 & 7 & 8 \\ 3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 \\ 5 & 6 & 6 & 7 & 6 & 7 & 7 & 8 \\ 5 & 6 & 6 & 7 & 6 & 7 & 7 & 8 \\ 7 & 8 & 8 & 9 & 8 & 9 & 9 & 10 \end{pmatrix}$$

Sve pomoćne matrice koje su eventualno kreirane u funkciji moraju biti uklonjene prije izlaska iz funkcije (inače im nećemo moći pristupiti niodakle i imaćemo curenje memorije). U slučaju da je $n=1$, rezultirajuća matrica treba biti ista kao i izvorna. U slučaju da parametar n ima vrijednost manju od 1, funkcija baca izuzetak tipa "`domain_error`" uz prateći tekst "Besmislen parametar".

Pored ove funkcije, treba također proširiti funkciju "`IspisiMatricu`" s dodatna dva parametra nazvana "`sirina`" i "`treba_brisati`". Parametar "`sirina`" je cijeli broj koji određuje širinu ispisa, odnosno broj znakova koje će zauzeti element pri ispisu (njega zapravo treba proslijediti funkciji "`cout.width`" ili manipulatoru "`setw`"). Ovaj parametar treba da ima podrazumijevanu vrijednost 4. Drugi parametar "`treba_brisati`" je tipa "`bool`". Ukoliko ovaj parametar ima vrijednost "`true`", funkcija treba da po obavljenom ispisu oslobodi prostor zauzet matricom koja joj je proslijeđena kao parametar, u suprotnom ne treba da radi ništa po tom pitanju. Ovim se omogućava da možemo zadavati pozive poput

```
IspisiMatricu(ZbirMatrica(a, b), 5, true);
```

tako da se oslobađanje memorije koju je zauzela pomoćna matrica koja predstavlja zbir matrica može obaviti bez korištenja pomoćne promjenljive (kao što smo morali u izvornom programu prikazanom na predavanjima). Pri tome, parametar "`treba_brisati`" treba imati podrazumijevanu vrijednost "`false`", tako da ga ne moramo navoditi ukoliko nam brisanje ne treba.

Napisane funkcije testirajte u glavnom programu na primjeru matrica cijelih brojeva čije dimenzije i elemente unosi korisnik putem tastature. Za test funkcije "`ProsiriPremaFunkcijama`" sa tastature se također unosi i broj n, a za mapu koristite mapu iz prethodno navedenog primjera. Pri tome, kako dimenzije rezultatnte matrice veoma brzo (eksponencijalno) rastu s porastom n, prilikom testiranja koristite male vrijednosti n i matrice malog formata, recimo 2×2). Predvidite hvatanje svih izuzetaka koji bi eventualno mogli nastupiti.

Napomena: U ovom programu je *izuzetno lako* napraviti curenje memorije (konkretnije, u funkciji "`ProsiriPremaFunkcijama`"). Dobro razmislite šta radite, i ne oslanjate se nasumice ni na kakve "testere curenja memorije" i slična automatska pomagala, jer u suprotnom nećete dobro ovladati tehnikama upravljanja memorijom (to i jeste glavni cilj ovog zadatka). Kad shvatite koliko treba razumijevanja da se u ovom zadatku izbjegne curenje memorije, tek tada ćete *zaista cijeniti destruktore*, koji ovakve probleme rješavaju praktično automatski (doduše, pametni pokazivači bi također automatski riješili problem, ali u ovom zadatku ih ne smijete koristiti, slično kao što na polaganju vozačkog ispita ne možete polagati na automobilu s automatskim mjenjačem).

Napomena 2: Zadaci principijelno slični ovom zadatku su se javljali kao zadaci za zadaću u prethodnim generacijama. Naravno, možete "pođoniti" neki od tih zadataka (uz odgovarajuće modifikacije), ili Vam ga može uraditi neko drugi, za pare ili ne (kao što uostalom nekima od Vas drugi rade i ostale zadatke). Ali da znate, što bi rekao Balašević, *neko to od gore vidi sve*, i prije ili kasnije to će Vam se od glavu razbiti. Izuzetno je važno da barem ovaj zadatak *zaista uradite sami* (naravno, samostalno iste trebali uraditi i sve ostale zadatke, ali je izuzetno važno da baš ovaj zadatak ne prepišete ni po koju cijenu). Ukoliko ne želite samostalno da uradite ovaj zadatak, *radije ga nemojte ni raditi, ni predavati*.

4. Jedne sušne godine, količina vode u rezervoarima iz kojih se snabdijeva grad bila je tako loša da je direktor gradske uprave vodododa i kanalizacije gospodin Izmet Fekalović bio prisiljen da uvede redukcije vode u pojedinim dijelovima grada. Grad je podijeljen u gradske distrikte koji su numerirani redom od 1 do N. Plan je da svaki M-ti distrikt počev od disktrikta 1 redom bude stavljen na redukciju, nakon čega on ne dolazi u konkurenciju za ponovno stavljanje na redukciju sve dok svaki distrikt ne bude barem jednom stavljen na redukciju. Pri tome se podrazumijeva da iza posljednjeg distrikta ponovo dolazi distrikt 1, tako da se razbrajanje obavlja ciklično. Nakon što svi distrikti dođu na red, postupak se ponavlja ispočetka. Na primjer, ukoliko u gradu ima 10 distrikata (numeriranih od 1 do 10) i ukoliko je M = 4, redoslijed redukcija je 1, 5, 9, 4, 10, 7, 6, 8, 3 i 2 (nacrtajte sliku), nakon čega ponovo započinje isti ciklus redukcija.

   Predložena strategija je očigledno vrlo pravična. Međutim, vlasnica najekskluzivnijeg restorana u gradu, gospođa Splačina Nejestivović, želi da distrikt K u kojem se nalazi njen restoran bude posljednji stavljen na redukciju, da ne bi došlo do remećenja rada restorana u kritično vrijeme kada se planira dolazak komisije koja treba da ocijeni da li restoran zaslužuje Michelinovu zvjezdicu. Zbog toga gospođa Splačina planira da iskoristi svoja lična poznanstva s gospodinom Izmetom sa ciljem da on odabere takav broj M da distrikt K bude posljednji stavljen na redukciju.

   Potrebno je napraviti program koji će gospođi Splačini pomoći da njen restoran dobije Michelinovu zvjezdicu (koju će sigurno dobiti ukoliko je redukcija vode ne omete). U programu treba implementirati dvije funkcije, "`Razbrajanje`" i "`OdabirKoraka`". Funkcija "`Razbrajanje`" prima kao parametre broj distrikta N i korak razbrajanja M, a kao rezultat daje vektor koji redom sadrži blojeve distrikata koji se stavljaju na redukciju, poredane u redoslijedu vršenja redukcija. Na primjer, ukoliko je N = 10 i M = 4, ova funkcija vraća vektor čiji su elementi redom 1, 5, 9, 4, 10, 7, 6, 8, 3 i 2. Pri tome, funkcija treba biti zasnovana na bibliotečkom tipu "`list`", koji se često primjenjuje upravo za rješavanje problema koji su srodni opisanom problemu. Naime, funkcija "`Razbrajanje`" će prvo napuniti listu rednim brojevima od 1 do N. Nakon što je formirana lista, vrši se kretanje kroz listu, polazeći od početka (prvog distrikta), pri čemu se nakon svakih M napravljenih koraka odstranjuje onaj element iz liste na kojem se trenutno nalazimo (čime se odgovarajući distrikt efektivno odstranjuje iz razmatranja), a njegov redni broj se smješta u izlazni vektor, nakon čega prelazimo na sljedeći element. Pri tome, kad god dostignemo kraj liste, vraćamo se ponovo na početak liste, čime zapravo simuliramo "kružnu" listu (koja nije direktno podržana kao bibliotečki tip podataka). Postupak se ponavlja sve dok se ne eliminira svih N elemenata, nakon čega vraćamo vektor u kojem smo zapamtili redoslijed odstranjivanja.

Već je rečeno da u programu pored funkcije "`Razbrajanje`", treba također implementirati i funkciju "`OdabirKoraka`". Ova funkcija kao parametre prima ukupan broj distrikta N i broj odabranog distrikta K, a kao rezultat daje korak razbrajanja M koji treba uzeti da bi posljednji distrikt stavljen na redukciju bio upravo distrikt K. Ukoliko takvih koraka ima više, funkcija vraća najmanji mogući korak, a ukoliko slučajno takva vrijednost koraka ne postoji, funkcija kao rezultat vraća 0 (postavljaču zadatka nije poznato može li se ovo ikada desiti, ali predvidimo i ovo za svaki slučaj). Rad ove funkcije zasniva se na pozivanju funkcije "`Razbrajanje`" za razne vrijednosti M sve dok se ne pronađe potrebni korak. Napisane funkcije demonstrirajte u testnom programu koji za zadane vrijednosti N i K ispisuje traženi korak razbrajanja koji gospođa Splačina treba dogovoriti s gospodinom Izmetom da bi se postigao željeni cilj.

5. Riješite ponovo prethodni zadatak, ali tako što ćete za realizaciju funkckije "`Razbrajanje`" umjesto bibliotečki definiranog tipa podataka "`list`" koristiti ručno kreiranu povezanu listu čvorova, bez upotrebe ikakvih bibliotečki definiranih tipova. To ćete izvesti ovako. Prvo ćete na globalnom nivou definirati čvornu strukturu "`Distrikt`" koja predstavlja jedan element liste, odnosno jedan gradski distrikt (ona mora biti definirana na globalnom nivou, da bi bila dostupna za potrebe autotestiranja, iako bi s aspekta samog zadatka mogla biti definirana i lokalno unutar funkcije). Ona treba sadržavati polje "`broj_distrikta`" tipa "**`int`**" i polje "`sljedeci`" koje je po tipu pokazivač na strukturu tipa "`Distrikt`". Polje "`broj_distrikta`" sadržavaće broj distrilta, dok će polje "`sljedeci`" pokazivati na sljedeći distrikt. Funkcija "`Razbrajanje`" treba kreirati povezanu listu distrikata (tj. čvorova tipa "`Distrikt`") čiji će brojevi biti postavljeni redom na vrijednosti od 1 do N, pri čemu će svaki čvor sadržavati pokazivač koji pokazuje na distrikt sa narednim brojem, osim posljednjeg čvora koji će pokazivati ponovo na prvi distrikt, čime se zapravo kreira krug distrikata. Nakon što je formirana tražena kružna lista, vrši se kretanje kroz listu, polazeći od prvog distrikt, pri čemu se nakon svakih M napravljenih koraka odstranjuje onaj distrikt iz liste na kojem se trenutno nalazimo, a njegov broj smješta se u izlazni vektor (sâmo odstranjivanje pri tome izvodimo manipuliranjem s pokazivačima koje predstavljaju veze između čvorova. tako što prvo "isključimo" sporni čvor iz lanca čvorova, a zatim ga i fizički uklonimo pomoću operatora "**`delete`**"). Postupak se ponavlja dok se ne eliminira svih N distrikata, nakon čega vraćamo vektor u kojem smo zapamtili redoslijed eliminiranja. U svim ostalim detaljima (osim u načinu realizacije funkcije "`Razbrajanje`"), ovaj program treba biti identičan programu iz prethodnog zadatka. Dobro pazite da nigdje ne napravite curenje memorije!

6. Riješite ponovo prethodni zadatak, ali tako što će će se svugdje umjesto običnih koristiti pametni pokazivači. Vrijede iste napomene kao i u prethodnom zadatku. Posebno trebate paziti da nigdje ne napravite curenje memorije, koje u ovom zadatku lako možete napraviti (ako ste neoprezni), bez obzira što se u njemu koriste isključivo pametni pokazivači!

NAPOMENA: Sve eventualne nedorečenosti u postavkama zadataka biće razriješene putem javnih autotestova.