Zadaća 3.

Ova zadaća nosi ukupno 5 poena, pri čemu zadaci nose redom 0.9, 0.9, 0.8, 0.8, 1.2 i 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 nedjelja, 22. V 2022. 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 i daje neprirodno izlomljen grafik. Bolje bi bilo kada bi grafik funkcije bio "gladak" i da pri tom ne pravi nikakva neprirodna "izvijanja" ili oscilacije u dijelovima između čvorova interpolacije. Ovim problemom bavili su se brojni poznati naučnici, uključujući *Lagrangea*, *Newtona* i *Gaussa*, i pri tome postigli relativno zadovoljavajuće rezultate. mada ponekad praćene priličnim problemima. Dugo vremena jedna od najuspješnijih tehnika interpolacije bila je relativno složena tehnika poznata kao *interpolacija pomoću kubnih splajnova*. Međutim, posljednjih godina razvijena je jedna nova i jednostavna tehnika, slabo poznata na ovim prostorima, a koja se u većini primjena pokazala u većini stvari superiornom u odnosu na sve druge poznate tehnike. Ova tehnika, poznata pod nazivom *baricentrična racionalna interpolacija*, računa vrijednost funkcije f po formuli

$$f(x) = \frac{\displaystyle\sum_{i=1}^{n} \frac{w_i}{x - x_i} y_i}{\displaystyle\sum_{i=1}^{n} \frac{w_i}{x - x_i}}$$

gdje su w_i, $i = 1, 2, \ldots, n$ odgovarajući težinski koeficijenti. Striktno posmatrano , ovaj izraz za $f(x)$ nije definiran ukoliko je $x = x_i$ za neko $i = 1, 2, \ldots, n$. Međutim, lako se vidi da vrijedi $f(x) \to y_i$ za $x \to x_i$, tako da ima smisla smatrati da je $f(x_i) = y_i$. Drugim riječima, ovako konstruirana funkcija $f(x)$ uvijek prolazi kroz sve zadane čvorove interpolacije (x_i, y_i), $i = 1, 2, \ldots, n$, kakvi god da su težinski koeficijenti. Sada je ključna ideja da se odrede težinski koeficijenti tako da se postigne zadovoljavajuće ponašanje funkcije $f(x)$ van čvorova interpolacije. Ovaj problem matematičari istražuju već dugi niz godina. Tako je 1988. godine *J. P. Berrut* pokazao da se relativno dobri rezultati mogu dobiti već upotrebom sasvim jednostavnih težinskih koeficijenata $w_i = (-1)^{i-1}$, $i = 1, 2, \ldots, n$. Međutim, 2006. godine su *M. S. Floater* i *K. Horman* dobili izvanredne rezultate upotrebom težinskih koeficijenata koji se računaju prema formuli

$$w_i = \sum_{k=\max\{1, i-d\}}^{\min\{i, n-d\}} (-1)^{k-1} \prod_{\substack{j=k \\ j \neq i}}^{k+d} \frac{1}{x_i - x_j} \qquad i = 1,2, \ldots, n$$

Ovdje je d slobodni parametar, tzv. *red interpolacije*, koji mora biti izabran tako da je $0 \leq d < n$, a obično se uzima $d \ll n$ (Berrutovi koeficijenti mogu se dobiti kao specijalni slučaj iz ove formule za $d = 0$). Porastom d kvalitet interpolacije tipično raste (tj. greška interpolacije izvan čvorova se smanjuje), ali obično do određene granice, nakon čega kvalitet interpolacije počne opadati (mogu se pojaviti vidnije oscilacije između čvorova). Okvirno vrijedi da što su čvorovi gušće raspoređeni, vrijednost d smije biti veća. Obično se ne koriste vrijednosti d veće od 6.

Vaš zadatak je da napravite funkciju za podršku za baricentričnu racionalnu interpolaciju. Ova funkcija se treba zvati "`BaricentricnaInterpolacija`", a imaće dvije verzije. Prva verzija ima dva parametra, pri čemu je prvi parametar vektor uređenih parova (tj. objekata tipa "`std::pair`") realnih brojeva, koji predstavljaju čvorove interpolacije, a drugi parametar je cijeli broj (tipa "`int`") koji predstavlja red interpolacije d. Kao rezultat, ova funkcija treba vratiti novu funkciju sa jednim realnim argumentom x koja obavlja traženu baricentričnu racionalnu interpolaciju za

zadanu vrijednost x (to je zapravo funkcija f iz prethodnog teorijskog razmatranja). Na primjer, ukoliko se izvrši naredba

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

tada će "`f`" biti funkcija dobijena baricentričnom racionalnom interpolacijom reda $d = 2$ 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 $f(2.5) = 5.425$), možemo izvršiti recimo naredbu

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

U slučaju da red interpolacije d nije u opsegu od 0 do broja čvorova, treba baciti izuzetak tipa "`domain_error`" sa pratećim tekstom "Nedozvoljen red". Također, ukoliko među čvorovima interpolacije postoje čvorovi koji imaju identične x-koordinate (što nije dozvoljeno), treba baciti isti izuzetak, ali uz prateći tekst "Neispravni cvorovi".

Druga verzija funkcije "`BaricentricnaInterpolacija`" služi za aproksimaciju neke već postojeće funkcije uz pomoć baricentrične racionalne interpolacije. Prvi parametar ove funkcije je neka realna funkcija jednog realnog argumenta (ili bilo šta što se može pozvati sa jednim realnin argumentom i daje rezultat koji se može interpretirati kao realan broj) i on predstavlja funkciju koju aproksimiramo. Drugi, treći i četvrti parametar ćemo nazvati redom x_{min}, x_{max} i Δx, dok je peti parametar red interpolacije d. 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 baricentričnom racionalnom interpolacijom i da vrati tako konstruisanu funkciju kao rezultat. 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 $f(x) = x^2 + \sin x$. 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 red interpolacije: 2
Unesite argument (ili "kraj" za kraj): 2.5
f(2.5) = 5.425
Unesite argument (ili "kraj" za kraj): 4
f(4) = 4
Unesite argument (ili "kraj" za kraj): 5.32
f(5.32) = 1.47372
Unesite argument (ili "kraj" za kraj): kraj

Odaberite opciju (1 - unos cvorova, 2 - aproksimacija): 2
Unesite krajeve intervala i korak: 0 1 0.1
Unesite red interpolacije: 2
Unesite argument (ili "kraj" za kraj): 0.3
f(0.3) = 0.38552 fapprox(0.3) = 0.38552
Unesite argument (ili "kraj" za kraj): 0.57
f(0.57) = 0.864532 fapprox(0.57) = 0.864515
Unesite argument (ili "kraj" za kraj): 5
f(5) = 24.0411 fapprox(5) = 24.289
Unesite argument (ili "kraj" za kraj): kraj
```

Kao što je vidljivo, 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 funckije i aproksimativna vrijednost dobijena interpolacijom). 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. Ukoliko dođe do bacanja izuzetka, treba ispisati odgovarajući tekst izuzetka i odmah prekinuti program.

NAPOMENA: Nije nimalo mudra ideja preračunavati težinske koeficijente w_i, $i = 1, 2, \ldots, n$ unutar sâme funkcije koja se vrati kao rezultat "`BaricentricnaInterpolacija`". Naime, na taj način će se ti koeficijenti svaki put iznova računati kad god se tako vraćena funkcija pozove, što je neprihvatljiv utrošak vremena ukoliko je n veliko. Umjesto toga, te koeficijente trebalo bi izračunati unutar funkcije "`BaricentricnaInterpolacija`", smjestiti ih negdje, i "zarobiti" ih u funkciju koja se vraća kao rezultat. Na taj način će se pri svakom pozivu vraćene funkcije koristiti koeficijenti koji su predhodno izračunati, što je velika ušteda u vremenu.

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 (obično brojevi stranica) na kojima se taj pojam javlja u tekstu. U ovom zadatku ćete trebati uraditi nešto slično tome. Prvo ćete napraviti funkciju "`KreirajIndeksPojmova`" koja kao parametar prima neki string (tipa "`string`") 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 cijelih brojeva. Vrijednosti ključnih polja predstavljaju različite riječi pronađene u analiziranom tekstu, dok su odgovarajuće pridružene vrijednosti skupovi čiji su elementi pozicije (tj. indeksi unutar razmatranog stringa) na kojima se odgovarajuća riječ nalazi unutar razmatranog stringa. Na primjer, ukoliko se kao parametar funkciji ponudi string "abc qwe stsda abc abc dhi qwe hrkw dhi", funkcija treba da vrati mapu u kojoj se nalazi 5 parova (toliko ukupno ima različitih riječi u stringu), čija su ključna polja stringovi "abc", "dhi", "hrkw", "qwe" i "stsda" (riječi koje se nalaze u stringu), dok su odgovarajuća ključna polja skupovi {0, 14, 18}, {22, 35}, {30}, {4, 26} i {8} (riječ "abc" se nalazi na pozicijama 0, 14 i 18 u razmatranom tekstu, riječ "dhi" 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 baca izuzetak tipa "`logic_error`" sa propratnim tekstom "Pojam nije nadjen" u slučaju da data riječ nije nađena u indeksu pojmova. 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 (bez ikakvih razmaka). Na primjer, za indeks pojmova kreiran na osnovu stringa iz prethodnog primjera, ova funkcija bi trebala proizvesti ispis poput sljedećeg:

```
abc: 0,14,18
dhi: 22,35
hrkw: 30
qwe: 4,26
stsda: 8
```

String koji se analizira može 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 malih 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, nakon čega se na ekranu ispisuje kreirani indeks pojmova. Potom program ulazi u petlju u kojoj se za svaku riječ unesenu sa tastature ispisuje indeks pozicija 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!". Program treba da prekine rad kada korisnik unese znak "." (tačka) sa tastature. Dijalog između korisnika i programa trebao bi izgledati poput sljedećeg:

```
Unesite tekst: abc qwe stsda abc abc dhi qwe hrkw dhi
abc: 0,14,18
dhi: 22,35
hrkw: 30
qwe: 4,26
stsda: 8
Unesite rijec: abc
0 14 18
Unesite rijec: hrkw
30
Unesite rijec: xyzzy
Unesena rijec nije nadjena!
Unesite rijec: .
```

3. Neka je **A** neka kvadratna matrica. Izraz oblika

$$\mathrm{P}_n(\mathbf{A}) = p_0\mathbf{I} + p_1\mathbf{A} + p_2\mathbf{A}^2 + \ldots + p_n\mathbf{A}^n = \sum_{k=0}^{n} p_k\mathbf{A}^k$$

gdje su $p_k, k = 0, 1, \ldots, n$ neki realni koeficijenti naziva se *matrični polinom*. Ovdje je **I** jedinična matrica istog formata kao i matrica **A**, dok je $\mathbf{A}^k$ klasični k-ti stepen matrice, tj. produkt matrice **A** k puta sa samom sobom (razumije se da zbog toga **A** mora biti kvadratna matrica), uz dodatnu konvenciju $\mathbf{A}^0 = \mathbf{I}$. Matrični polinomi imaju veliku primjenu u raznim oblastima nauke i tehnike, posebno u teoriji automatskog upravljanja i robotici. Vaš zadatak je da dopunite program za demonstraciju generičke strukture "`Matrica`" obrađen na Predavanju 8_b sa dvije nove funkcije "`ProduktMatrica`" i "`MatricniPolinom`". Funkcija "`ProduktMatrica`" prima dvije matrice kao parametre (pri čemu su matrice definirane kao odgovarajuće generičke strukture tipa "`Matrica`") i vraća njihov produkt kao rezultat, odnosno baca izuzetak tipa "`domain_error`" uz prateći tekst "Matrice nisu saglasne za mnozenje" ukoliko matrice nisu saglasne za množenje. Funkcija "`MatricniPolinom`" prima matricu **A** kao prvi parametar, a kao drugi parametar vektor realnih brojeva koji sadrži koeficijente $p_k, k = 0, 1, \ldots, n$, dok kao rezultat vraća vrijednost matričnog polinoma $\mathrm{P}_n(\mathbf{A})$ (za potrebe realizacije ove funkcije trebate koristiti već napisanu funkciju "`ProduktMatrica`"). U slučaju da matrica nije kvadratna, funkcija treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Matrica mora biti kvadratna", s obzirom da su matrični polinomi definirani samo za kvadratne matrice. U slučaju da je drugi parametar prazan vektor, funkcija treba kao rezultat da vrati nul-matricu **O** (tj. matricu čiji su svi elementi nule) istog formata kao i matrica **A**.

Pored ove dvije funkcije, treba također proširiti funkciju "`IspisiMatricu`" sa dodatna dva parametra nazvana "`preciznost`" i "`treba brisati`". Parametar "`preciznost`" je cijeli broj koji određuje preciznost ispisa, odnosno broj tačnih cifara pri ispisu (njega zapravo treba proslijediti funkciji "`cout.precision`" ili manipulatoru "`setprecision`"). Ovaj parametar treba da ima podrazumijevanu vrijednost 6. 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), 10, 5, true);
```

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

Napisane funkcije testirajte u glavnom programu koji sa tastature prvo traži da se unese dimenzija matrice (dovoljno je unijeti samo jednu dimenziju, s obzirom da će matrica uvijek biti kvadratna), zatim elementi matrice (pozivom funkcije "`UnesiMatricu`", zadajući "A" kao ime matrice), zatim red polinoma n, te na kraju koeficijenti polinoma, počev od slobodnog člana nadalje. Po obavljenom unosu, program treba da ispiše matricu dobijenu izračunavanjem matričnog polinoma koristeći širinu ispisa od 10 mjesta i preciznošću od 6 tačnih cifara. Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

```
Unesite dimenziju kvadratne matrice: 2
Unesite elemente matrice A:
A(1,1) = 3
A(1,2) = 1
A(2,1) = 0
A(2,2) = -2
Unesite red polinoma: 3
Unesite koeficijente polinoma: 3 -2 0 1
        24         5
         0        -1
```

Prikazani rezultat slijedi iz računa

$$3\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}-2\begin{pmatrix}3 & 1\\ 0 & -2\end{pmatrix}+0\begin{pmatrix}9 & 1\\ 0 & 4\end{pmatrix}+1\begin{pmatrix}27 & 7\\ 0 & -8\end{pmatrix}=\begin{pmatrix}24 & 5\\ 0 & -1\end{pmatrix}$$

U testnom programu ne može doći do bacanja izuzetka (osim eventualno usljed nedostatka memorije), tako da ne trebate predvidjeti hvatanje izuzetaka (to ne znači da testovi neće testirati da li funkcije koje trebaju bacati izuzetke ispravno bacaju ono što treba kad treba). Također, dobro pazite da nigdje ne dođe do curenja memorije, ni pod kakvim okolnostima (u suštini, to je i *osnovni cilj zadatka*).

NAPOMENA 1: U ovom programu je *iznimno lako* napraviti curenje memorije (konkretno, u funkciji "`MatricniPolinom`"). Dobro razmislite šta radite, i ne oslanjate se nasumice ni na kakve "testere curenja memorije" i slična pomagala, jer u suprotnom nećete 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.

NAPOMENA 2: Zadaci slični ovom zadatku su se javljali povremeno 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, niti predavati*.

4. Skupina od n djece želi se razvrstati u k timova za neku igru. Timovi bi trebali biti iste veličine, ali to nije uvijek moguće, osim ako je n tačno djeljivo sa k. Zbog toga su odlučili da prvih $\mathrm{mod}(n,k)$ timova ima $\lfloor n/k \rfloor + 1$ članova, a preostali timovi $\lfloor n/k \rfloor$ članova (ovdje $\mathrm{mod}(n,k)$ označava ostatak pri dijeljenju n sa k, a $\lfloor \ldots \rfloor$ označava cijeli dio). Ova strategija garantira da će se broj članova u pojedinim timovima razlikovati najviše za 1. Recimo, ukoliko je $n = 22$ i $k = 5$, tada je $\mathrm{mod}(n,k) = 2$ i $\lfloor n/k \rfloor = 4$, tako da će prva dva tima imati 5 članova a preostala 3 tima 4 člana. Samo razvrstavanje obavlja se na sljedeći način. Sva djeca se poredaju u krug, nakon čega se obavlja razbrajanje slično "eci peci pec" sistemu, samo uz modifikacije koje će biti opisane, da razbrajanje bude interesantnije. Razbrajanje počinje od nekog djeteta, koje odmah ide u prvi tim i "ispada iz kruga". Nakon toga se odbrojava u smjeru kazaljke na satu, pri čemu se broji onoliko koraka koliko slova ima u imenu djeteta koje je upravo ispalo iz kruga. Novo dijete na kojem se odbrojavanje zaustavi također ide u prvi tim i ispada iz kruga, a broj slova njegovog imena koristi se kao broj koraka za naredno razbrajanje. Postupak se ponavlja dok se ne formira prvi tim, nakon čega se nastavlja dalje po istom sistemu, samo što djeca koja ispadaju iz kruga idu u drugi tim, itd. sve dok se ne formiraju svi timovi. Uzmimo kao primjer da imamo desetero djece čija su imena Damir, Ana, Muhamed, Marko, Ivan, Mirsad, Nikolina, Alen, Jasmina i Merima i da je potrebno formirati 3 tima (koji će imati redom 4, 3 i 3 člana). Neka su djeca stala u krug baš navedenim redom i nek razbrajanje počinje od Damira. Damir odmah odlazi u prvi tim, nakon čega idemo 5 koraka u smjeru kazaljke na satu. Zaustavljamo se na Mirsadu koji također odlazi u prvi tim, i idemo 6 koraka dalje (u smjeru kazaljke na satu). Dolazimo do Muhameda, koji odlazi u prvi tim, nakon čega idemo 7 koraka dalje. Time dolazimo do Ane, koja također odlazi u prvi tim, čime je prvi tim kompletiran. Nakon Ane, idemo 3 koraka dalje, te se zaustavljamo na Nikolini

koja odlazi u drugi tim. Nakon toga, preskačući detalje, naredna dva člana koja idu u drugi tim su Merima i Ivan, dok Alen, Marko i Jasmina (tim redom) odlaze u treći tim, čime je postupak gotov.

Vaš zadatak je da napravite funkciju "`Razvrstavanje`" koja simulira gore opisani postupak. Parametar funkcije je vektor stringova koja sadrži imena djece koja se razvrstavaju, te broj timova koji želimo formirati (tipa "`int`"). Imena su poredana tačno onim redom kako će djeca stajati u krug (u smjeru kazaljke na satu), a razbrajanje počinje od djeteta čije je ime prvo u vektoru. Kao rezultat, funkcija vraća vektor čiji su elementi skupovi stringova, pri čemu svaki skup odgovara jednom formiranom timu. Pri tome, funkcija mora biti zasnovana na tipu "`list`", koji se često primjenjuje upravo za rješavanje problema srodnih opisanom problemu (zadatak se *ne priznaje* ukoliko se ovaj zahtjev ne ispoštuje). Naime, funkcija "`Razvrstavanje`" će prvo sva imena iz vektora iskopirati u neku listu. Nakon što je formirana lista, vrši se kretanje kroz listu, polazeći od početka, pri čemu kad god dođemo na element koji odgovara djetetu koje napušta krug, taj element kopiramo u odgovarajući skup koji odgovara timu koji trenutno formirano, nakon čega ga odstranjujemo iz liste (čime se to dijete efektivno "odstranjuje iz kruga"), Nakon toga, prelazimo na sljedeći element. Pri tome, kad god dostignemo kraj liste, vraćamo se ponovo na početak, čime simuliramo "kružnu" listu, odnosno listu čiji je kraj spojen sa početkom. Postupak se ponavlja dok se ne razvrstaju sva djeca u timove, nakon čega vraćamo vektor koji sadrži formirane timove kao rezultat. Kao primjer, ukoliko je ulazni vektor {"Damir", "Ana", "Muhamed", "Marko", "Ivan", "Mirsad", "Nikolina", "Alen", "Jasmina", "Merima"} i ukoliko treba formirati 3 tima, izlazni vektor treba sadržavati redom skupove {"Ana", "Damir", "Mirsad", "Muhamed"}, {"Ivan", "Merima", "Nikolina"} i {"Alen", "Jasmina", "Marko"}. Neka Vas ne zbunjuje to što imena u skupovima nisu u redoslijedu kako su pojedina djeca ulazila u timove. Naime, elementi skupova se automatski čuvaju u sortiranom poretku, što nam nije smetnja, jer sam redoslijed elemenata u timu nije bitan (bitno je samo koja djeca čine jedan tim). Ukoliko je broj timova manji od 1 ili veći od broja djece, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Razvrstavanje nemoguce".

Napisanu funkciju iskoristite u testnom programu koji formira dijalog između korisnika i programa poput sljedećeg (dijalog je jasan sam po sebi, tako da dodatna objašnjenja nisu potrebna):

```
Unesite broj djece: 10
Unesite imena djece:
Damir
Ana
Muhamed
Marko
Ivan
Mirsad
Nikolina
Alen
Jasmina
Merima
Unesite broj timova: 3
Tim 1: Ana, Damir, Mirsad, Muhamed
Tim 2: Ivan, Merima, Nikolina
Tim 3: Alen, Jasmina, Marko
```

U slučaju da dođe do bacanja izuzetka, treba ispisati tekst izuzetka i prekinuti program.

Dopušteno je da imena imaju više riječi, odnosno legalna su imena poput Abdu Samed, Ana Marija, Bin Laden (o imenu Bin Laden pogledati pod Bin Laden Krupić), Isus Hristos Simeun Stanko (ima i ovoga u Bosni) ili David Vid Viktorije Dimitrije Sebastijan Ernest (tako se zove sin Darka Rundeka). Međutim, treba imati u vidu da se samo slova uzimaju u obzir pri odabiru koraka razbrajanja (razmaci se ignoriraju), tako da se za ovih pet imena smatra da imaju respektivno dužine 9, 10, 8, 23 i 42. Isto vrijedi za znakove koji nisu niti slova niti cifre, tako da ime poput Mubarek-Kerim ima dužinu 12 (a ne 13). Što se tiče cifara, zakoni baš ne gledaju blagonaklono na upotrebu cifara u imenima (jedan švedski par se bori da djetetu da ime Brfxxccxxmnpccccllllmmnprxvclmnckssqlbb11116 tvrdeći da se to izgovara kao Albin), ali to ćemo ovdje dopustiti (tretirajući cifre identično kao i slova), tako da ćemo smatrati da dužina imena 2PAC iznosi 4.

5. Riješite ponovo prethodni zadatak, ali tako što ćete za realizaciju funkckije "`Razvrstavanje`" umjesto bibliotečki definiranog tipa podataka "`list`" koristiti ručno kreiranu povezanu listu čvorova (dakle, bez upotrebe ikakvih bibliotečki definiranih tipova). To ćete izvesti ovako. Prvo ćete definirati čvornu strukturu "`Dijete`" koja predstavlja jedan element liste, odnosno jedno dijete koje se razvrstava. Ona treba sadržavati polje "`ime`" tipa "`string`" i polje "`sljedeci`" koje je po tipu pokazivač na strukturu tipa "`Dijete`". Polje "`ime`" sadržavaće ime djeteta, dok će polje "`sljedeci`" pokazivati na sljedeće dijete u krugu. Funkcija "`Razvrstavanje`" treba kreirati povezanu listu djece (tj. čvorova tipa "`Dijete`"), pri čemu će svako dijete pokazivati na sljedeće dijete u krugu, osim posljednjeg djeteta (tj. posljednjeg čvora) koje će pokazivati ponovo na prvo dijete, čime se zapravo kreira krug djece (takve povezane liste u kojima posljednji čvor u listi pokazuje nazad na prvi čvor nazivaju se kružne ili cirkularne liste). Nakon što je formirana tražena lista, vrši se kretanje kroz listu, polazeći od prvog čvora, pri čemu se svaki put kada dođe vrijeme za odstranjivanje djeteta iz kruga odstranjuje onaj čvor iz liste na kojem se trenutno nalazimo, a ime koje se nalazilo u tom čvoru smješta se u skup koji odgovara timu koji u tom trenutku formiramo. Odstranjivanje se izvodi tako što se pokazivač "`sljedeci`" unutar čvora koji prethodi čvoru na kojem se trenutno nalazimo preusmjerava tako da ne pokazuje više na čvor na kojem se trenutno nalazimo nego na čvor koji slijedi iza njega (čime se čvor efektivno odstranjuje iz razmatranja, što odgovara uklanjanju djeteta iz kruga) i prelazimo na sljedeći čvor. Tom prilikom, pomoću operatora "**`delete`**" potrebno je izbrisati čvor koji odgovara djetetu koje je ispalo iz kruga, da ne zauzima više memoriju. Postupak se ponavlja dok se ne formiraju svi timovi, nakon čega vraćamo vektor formiranih timova. U svim ostalim detaljima (osim u načinu realizacije funkcije "`Razvrstavanje`"), ovaj program treba biti identičan programu iz prethodnog zadatka.

6. Riješite ponovo prethodni zadatak, ali tako što će polje "`sljedeci`" u čvornoj strukturi "`Dijete`" biti "pametni" umjesto običnog pokazivača. Suštinska razlika je što Vam ovaj put neće trebati operator "**`delete`**", jer će svaki čvor automatski nestati čim se isključi iz lanca, s obzirom da tada niko neće pokazivati na njega. Međutim, morate paziti da kada ostane samo jedan čvor, njegovo polje "`sljedeci`" će pokazivati na njega samog (tj. upravo na taj jedini preostali čvor). Ovu vezu obavezno morate raskinuti prije nego što napustite funkciju, inače će doći do curenja memorije (s obzirom da će taj čvor nastaviti da "čuva samog sebe" čak i kad nestanu drugi pokazivači koji su na njega pokazivali).