Programiranje i programsko inženjerstvo

Zadaci za masovne instrukcije (ispravljena verzija)

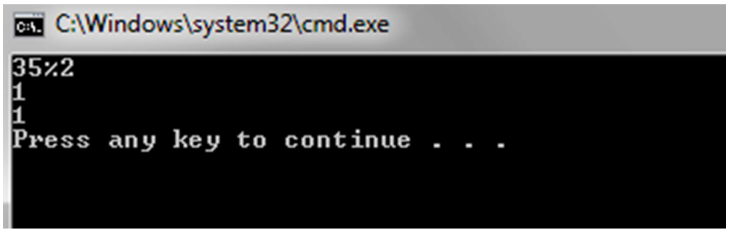
by Franko

24. studenti 2012.

1. Zadatak

Napravite mini kalkulator – unesite cijeli broj, potom računsku operaciju i potom ponovo cijeli broj. Na ekran ispišite rezultat operacije. Koristite samo četiri osnovne računske operacije i mod (%). Zadatak riješite uz pomoć:

a) Naredbe `if-else`
b) Naredbe `switch-case`




```c
#include <stdio.h>
int main ()
{
	int a,b;
	char op;

	scanf ("%d%c%d",&a, &op, &b);

	//prekinuti program ukoliko je drugi operand 0 u slucaju dijeljenja.
	if (((op == '/') || (op == '%')) && b == 0)
	{
		printf ("Nije moguce dijeliti broj sa nulom!\n");
		return -1;
	}

	if (op == '+')
		printf ("%d\n", a+b);
	else if (op == '-')
		printf ("%d\n", a-b);
	else if (op == '/')
		printf ("%f\n", (float)a/b);
	else if (op == '*')
		printf ("%d\n", a*b);
	else if (op == '%')
		printf ("%d\n", a%b);
	else
		printf ("Pogresan operator!\n");

	switch(op)
	{
	case '+' :
		printf ("%d\n", a+b);
		break;
	case '-':
		printf ("%d\n", a-b);
		break;
	case '/':
		printf ("%f\n", (float)a/b);
		break;
	case '*':
		printf ("%d\n", a*b);
		break;
	case '%':
		printf ("%d\n", a%b);
		break;
	default:
		printf ("Pogresan operator!\n");
		break;
	}
	return 0;
}
```

2. Zadatak

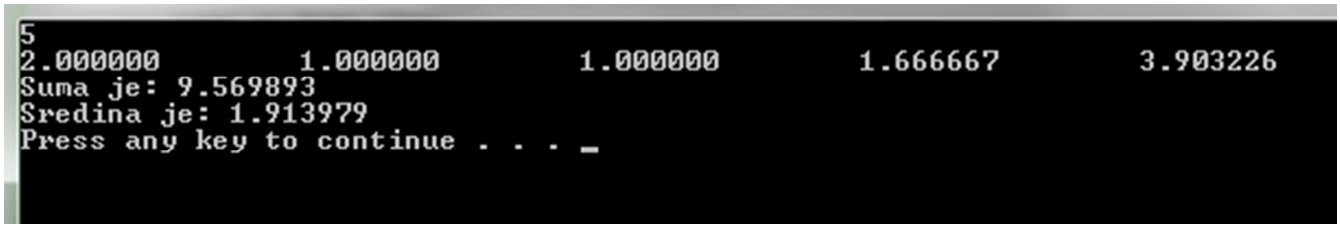
Unesite x. Potom izračunajte i na ekran ispišite:

a) Sumu i članove reda zadanog formulom:

$$\sum_{n=1}^{x} \frac{n! + 1}{2^n - 1}$$

b) Aritmetičku sredinu članova prethodnog reda.

Primjer:



Rješenje:

```c
#include <stdio.h>

int main() {
	int fakt = 1, pot = 1, x, i;
	float suma = 0, tren_clan, sredina;

	scanf("%d", &x);

	for(i = 1; i <= x; i++) {
		fakt *= i;
		pot *= 2;
		tren_clan = ((float)fakt+1)/((float)pot-1);
		suma += tren_clan;
		printf("%f\t", tren_clan);
	}

	printf("Suma je: %f\n", suma);
	sredina = suma/x;
	printf("Sredina je: %f\n", sredina);
	return 0;
}
```

Pogreška: U originalnim rješenjima je bila ovdje greška – faktorijeli i potencije su se povećavali NAKON izračuna trenutnog člana. To je za posljedicu imalo da se u idućem koraku za primjerice `i=2`, koristio `fakt` koji je bio povećan sa starim `i`, što je neispravno (faktorijel je uvijek „kasnio“ za jedan korak).

3. Zadatak

Učitajte troznamenkasti heksadekadski broj. Na ekran ispišite broj:

a) dekadski
b) oktalno
c) binarno

```c
#include <stdio.h>

int main ()
{
	char a,b,c;
	int dekadski = 0, oktalni = 0;
	int pom, mnozitelj,brojac,i;
	int binarni[100];

	scanf ("%c%c%c", &a, &b, &c);

	if (a >= 'A' && a <= 'F')
		dekadski = 256 * (10 + a - 'A');
	else if (a >= 'a' && a <= 'f')
		dekadski = 256 * (10 + a - 'a');
	else
		dekadski = 256 * (a - '0');
	if (b >= 'A' && b <= 'F')
		dekadski += 16 * (10 + b - 'A');
	else if (b >= 'a' && b <= 'f')
		dekadski += 16 * (10 + b - 'a');
	else
		dekadski += 16 * (b - '0');
	if (c >= 'A' && c <= 'F')
		dekadski += 10 + c - 'A';
	else if (c >= 'a' && c <= 'f')
		dekadski += 10 + c - 'a';
	else
		dekadski += c - '0';

	pom = dekadski;
	mnozitelj = 1;
	while (pom > 0)
	{
		oktalni += (pom%8) * mnozitelj;
		pom /= 8;
		mnozitelj *= 10;
	}

	pom = dekadski;
	brojac = 0;
	while (pom > 0)
	{
		binarni[brojac] = (pom%2);
		pom /=2;
		brojac++;
	}

	printf ("Dekadski: %d\n",dekadski);
	printf ("Oktalni: %d\n", oktalni);
	printf ("Binarni: ");
	for (i=brojac-1;i>=0;i--)
	{
		printf ("%d",binarni[i]);
	}
	printf ("\n");

	return 0;
}
```

4. Zadatak

Učitati prirodan troznamenkasti broj. Postupak učitavanja ponavljati sve dok nije učitan troznamenkasti broj.

a) Na ekran ispisati troznamenkasti broj kojem je svaka znamenka (pojedinačno) uvećana za 1 (ukoliko uvećavamo broj 9, njegov prvi uvećani broj je 0!), ali tako da on ostane troznamenkast. Primjerice broj 899 uvećavanjem postaje 900. 362 Postaje 473. Broj 988 nije moguće uvećati.
b) Na ekran ispisati prvi veći idući prosti broj. Ukoliko nije moguće pronaći prvi idući prosti troznamenkasti broj, na ekran ispisati odgovarajuću poruku. Primjerice 362, prvi veći prosti je 367.

```c
#include <stdio.h>

int main ()
{
	int broj;
	int i, jeProst = 0;
	int prvi, drugi, treci, noviBroj;

	do
	{
		scanf ("%d", &broj);
	} while (broj < 100 || broj > 999);

	prvi = broj % 10;
	drugi = broj / 10 % 10;
	treci = broj / 100 % 10;

	if (treci == 9)
	{
		printf ("Nije moguce uvecati broj.\n");
	}
	else
	{
		noviBroj = (treci + 1) * 100 + ((drugi + 1)%10) * 10 + (prvi + 1)%10;
		printf ("%d\n", noviBroj);
	}

	while (1)
	{
		broj++;
		if (broj > 999)
			break;

		i=2;
		jeProst = 1;

		while (i <= broj/2 + 1)
		{
			if (broj % i == 0)
			{
				jeProst = 0;
				break;
			}
			i++;
		}

		if (jeProst == 1)
			break;
	}
	if (jeProst == 1)
		printf("%d", broj);
	else
		printf("Nema veceg prostog broja\n");
	return 0;
}
```

Napomena: Prethodni zadatak je korisno proučiti iz više razloga. Na početku je primjenjena `do-while` petlja koja nam pokazuje jednostavan način za učitavanje vrijednosti dok nije učitana ispravna vrijednost.

Potom je napravljeno razbijanje broja na tri znamenke koristeći operaciju mod (%) i dijeljenjem broja. Algoritam je moguće poopćiti tako da radi za bilo koje znamenke i bilo koji interval. Primjer takvog algoritma je dan u nastavku:

```
#include <stdio.h>

int main ()
{
	int broj, znamenka;
	scanf ("%d", &broj);

	if (broj == 0)
	{
		znamenka = 0;
		printf ("%d\n", znamenka);
	} else if (broj < 0)
		broj *= -1;

	while (broj > 0)
	{
		znamenka = broj % 10;
		printf ("%d\n", znamenka);
		broj = broj / 10;
	}
	return 0;
}
```

Prvo ispitamo jeli broj samo jedna znamenka i to upravo nula. Ukoliko jest, ispisujemo na ekran nulu i to je efektivno kraj programa (while uvijet nije zadovoljen, pa se neće ni ulaziti u petlju). Ukoliko je broj negativan, učinimo ga pozitivnim. Na kraju uzastopno uzimamo uz pomoć mod zadnju znamenku te dijelimo broj. Znamenke se ispisuju sa desna na lijevo. Treba obratiti pažnju na to da smo sam broj u varijabli broj uništili uzastopnim dijeljenjem. U slučaju da nam sam broj treba nakon algoritma „razbijanja“, moramo ga prije algoritma pohraniti u pomoćnu varijablu (ili dijeliti tu pomoćnu varijablu, a očuvati sam broj u varijabli broj, što je bolje rješenje).

Vratimoi se na sam zadatak. Posljednje, u beskonačnoj petlji je dano rješenje koje traži prosti broj. Prost je broj onaj koji je djeljiv samo sa 1 i samim sobom. Algoritam se svodi na to da pretpostavimo za neki broj kandidat da jest prost broj (postavljanjem primjerice varijable `jeProst` na vrijednost 1). Potom tu pretpostavku pokušavamo oboriti sa dijeljenjem sa svim brojevima od 2 do `broj_kandidat/2+1`. Ukoliko nađemo broj koji dijeli bez ostataka u navedenom intervalu, očito se ne radi o prostom broju i postavljamo `jeProst` na nulu i prekidamo izvođenje unutarnje petlje (jer smo zaključili da broj nije prost!). Nakon toga ako smo naišli na prost broj, prekidamo beskonačnu petlju. Ako nismo petlja se nastavlja uvećavanjem broja kandidata sve dok broj kandidat ne postane tisuću (odnosno četveroznamenkast!).

Treba pojasniti i zašto smo odabrali interval do `broj_kandidat/2+1`. Očito je da broj 70 primjerice nije moguće podijeliti sa 60. Već samo sa najviše 35. Zato ne treba provjeravati brojeve koji su veći od polovice. Radi sigurnosti (da izbjegnemo nekakve probleme koje nam može stvoriti dijeljenje broja sa 2 koji nije djeljiv sa 2) smo dodali <= kako bismo podijeli sa 36 (iako u slučaju 70 nije potrebno, a vjerojatno ni u drugim brojevima – malo opreza nikada ne škodi).

Osnovni princip je moguće (pretpostavka, pa obaranje iste) je moguće primjeniti i na algoritam koji traži primjerice prvih 100 prostih brojeva (eventualno će u ovom slučaju trebati pripaziti sa rubnim slučajevima

ako nam broj kandidat kreće od 1, 2 – o tome razmislite sami gdje može biti problem) ali i u drugim algoritmima gdje je bolje pretpostaviti nešto pa to oboriti nego direktno dokazivati!

5. Zadatak

Naizmjenično učitavati parne i neparne prirodne brojeve sve dok se ne učita nešto što nije prirodan broj ili ako su uzastopce učitana dva parna ili dva neparna broja. Posljednji učitani broj (koji krši uvjet) ne uzimati u obzir! Na ekran ispisati:

a) Najveći parni i najmanji neparni broj. Ukoliko nije bio učitan ni jedan parni ili neparni broj dojaviti grešku (primjerice učitana su samo dva parna ili dva neparna broja ili 0 odmah na početku).
b) Definirajte racionalni broj kojem je najveći parni brojnik, a najveći neparni nazivnik. Skratite ta dva broja. Ispišite skraćene brojeve na ekran.
c) Ispitajte jeli paran broj djeljiv sa svojom najvećom znamenkom.

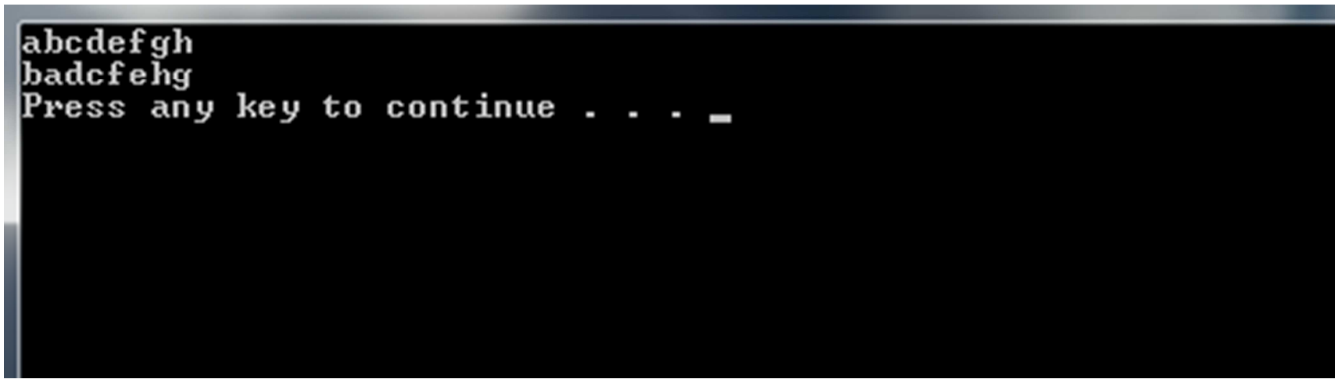
```c
#include <stdio.h>

int main ()
{
	int ucitano;
	int jeParan;
	int maxParan = -1, minNeparan = -1;
	int broj, jestPrvi = 1, iduci;
	int brojnik, nazivnik, djelitelj, pom, maxZnam;

	while (1)
	{
		scanf ("%d", &broj);

		if (broj <=0)
			break;

		if (jestPrvi == 1)
		{
			if (broj%2 == 0)
			{
				maxParan = broj;
				scanf ("%d", &broj);
				if ((broj%2 ==0) || (broj <= 0))
					break;
				else
				{
					minNeparan = broj;
					iduci = 0;
				}

			}
			else
			{
				minNeparan = broj;
				scanf ("%d", &broj);
				if ((broj%2==1) || (broj <=0))
					break;
				else
				{
					maxParan = broj;
					iduci = 1;
				}
			}
			jestPrvi = 0;
		}
```

```
		else
		{
			if (broj%2 != iduci)
				break;
			else
			{
				if (broj%2==0)
				{
					if (maxParan < broj)
						maxParan = broj;
					iduci = 1;

				}
				else
				{
					if (minNeparan > broj)
						minNeparan = broj;
					iduci = 0;

				}
			}
		}
	}

	if ((maxParan != -1) && (minNeparan != -1))
	{
		printf ("%d %d\n", maxParan, minNeparan);

		nazivnik = minNeparan;
		brojnik = maxParan;
		djelitelj = nazivnik;

		while (djelitelj > 0)
		{
			if ((brojnik % djelitelj == 0) && (nazivnik % djelitelj == 0))
				break;
			else
				djelitelj--;
		}

		printf ("%d / %d", brojnik / djelitelj, nazivnik / djelitelj);

		pom = maxParan;
		maxZnam = pom % 10;
		pom /= 10;
		while (pom > 0)
		{
			if (maxZnam < pom % 10)
				maxZnam = pom;
			pom /= 10;
		}

		if (maxParan % maxZnam == 0)
			printf ("\nParan broj je djeljiv sa svojom najvecom znamenkom!\n");
		else
			printf ("\nParan broj nije djeljiv sa svojom najvecom znamenkom!\n");

	}
	else
		printf ("Niste ucitali ispravne brojeve\n");
	return 0;

}
```

6. Zadatak

Učitajte polje znakova pomoću funkcije `scanf`. Može se pretpostaviti da neće učitano više od 100 znakova. Zamjenite svaka dva znaka koji čine par (primjerice, parove čine znakovi na indeksima 0 i 1, 2 i 3 itd.) Ukoliko je učitani broj znakova neparan, posljednji znak nema svog para, pa ga ne treba zamjenjivati već ostaviti kako jest. Rezultat ispišite na ekran.

Primjer:



Rješenje:

```c
#include <stdio.h>

int main ()
{
	char polje [100+1];
	int i;
	char pom;

	scanf ("%s", polje);

	i = 0;
	while (polje[i] != '\0')
	{
		if (polje [i+1] != '\0')
		{
			pom = polje [i];
			polje[i] = polje[i+1];
			polje[i+1] = pom;
		}
		else
			break;
		i+=2;
	}

	printf ("%s\n", polje);
	return 0;
}
```

Greška: U originalnom rješenju je nedostajao dio sa `else break;` Naime algoritam bez toga prekida samo ako se null znak nalazi na parnim mjestima, a ne i na neparnim. Dodavanje `else` dijela, algoritam prekida i ako se null znak nalazi na neparnom mjestu (neparnim indeksom – što je slučaj kod nizova neparne duljine).

7. Zadatak

Učitati rečenicu. Kraj rečenice je enter. Može se pretpostaviti da neće biti učitano više od 100 znakova. Za učitavanje koristiti funkciju `gets`.

a) Ispisati duljinu znakovnog niza
b) Ispisati frekvenciju pojavljivanja svakog pojedinog slova. Nije potrebno razlikovati velika i mala slova, već je dovoljno brojati
c) Sva velika slova pretvorite u mala, a mala u velika.

```
#include <stdio.h>

int main ()
{
	int i=0, duljina;
	char recenica [100+1];
	int frekvencija[26] = {0};

	gets (recenica);

	while (recenica [i] != '\0')
	{
		if (recenica[i] >= 'a' && recenica[i] <= 'z')
			frekvencija[recenica[i] - 'a']++;
		else if (recenica[i] >= 'A' && recenica[i] <= 'Z')
			frekvencija[recenica[i] - 'A']++;
		i++;
	}

	printf ("Ucitani niz je dugacak %d znakova\n", i);

	for (i=0;i<26;i++)
	{
		if (frekvencija[i] != 0)
			printf ("Slovo %c se pojavljuje %d puta\n", i+'A', frekvencija[i]);
	}

	duljina = i;
	for (i=0;i<duljina;i++)
	{
		if (recenica[i] >= 'a' && recenica[i] <= 'z')
			recenica[i] -= 'a' - 'A';
		else if (recenica[i] >= 'A' && recenica[i] <= 'Z')
			recenica[i] += 'a' - 'A';
	}

	printf ("\n%s\n", recenica);
	return 0;
}
```

Napomena: Treba obratiti pažnju na izraz:

```
recenica[i] -= 'a' - 'A';
```

Zbog niskog prioriteta operatora `-=` prvo se izračunava izraz sa desne strane, a potom se izračunati dio oduzme od sadržaja elementa `recenica[i]`. Ukrakto ovako:

```
recenica[i] = recenica[i] - ('a'-'A');
```

8. Zadatak

Unesite n, a potom n prirodnih brojeva. Može se pretpostaviti da neće biti zadano više od 50 brojeva. Nije potrebno provjeravati ispravnost učitanih brojeva. Ispišite na ekran onaj parni broj koji ima najveću sumu znamenaka te indeks u polju gdje se nalazi. Ukoliko dva broja imaju jednaku sumu, na ekran ispisati onaj sa manjim indeksom. Koristite simboličke konstante.

```c
#include <stdio.h>
#define MAX 50

int main ()
{
	int i,n,polje[MAX];
	int paranPostoji, broj, maxSuma, index, suma;

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

	paranPostoji = 0;
	for(i=0;i<n;i++)
	{
		if (polje[i] % 2 == 0)
		{
			broj = polje[i];
			suma = 0;
			while (broj > 0)
			{
				suma += broj % 10;
				broj /= 10;
			}

			if (paranPostoji == 0)
			{
				paranPostoji = 1;
				maxSuma = suma;
				index = i;
			}

			if (suma > maxSuma)
			{
				maxSuma = suma;
				index = i;
			}
		}
	}

	if (paranPostoji == 1)
		printf ("Broj sa najvecom sumom znamenaka jest %d na indeksu %d\n", polje[index], index);
	else
		printf ("Nije ucitan ni jedan paran broj!\n");

	return 0;
}
```

Napomena: Zbog grešaka i nepotrebno kompliciranja, cijeli ovaj zadatak je napisan isponova, te je sam zadatak redefiniran.

9. Zadatak

Učitajte dva znakovna niza. Prvi predstavlja rečenicu, a drugi riječ. Može se pretpostaviti da rečenica neće biti dulja od 100 znakova, a podniz dulji od 10. Koristite za učitavanje funkciju gets. Iz niza izbacite svako pojavljivanje podniza.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

int main ()
{
	char niz [MAX+1];
	char podniz [MAX+1];
	int pronadjeno;
	int i,j;
	int duljinaNiz, duljinaPodniz;

	gets (niz);
	gets (podniz);

	duljinaNiz = strlen(niz);
	duljinaPodniz = strlen(podniz);

	i=0;
	while(i<=duljinaNiz-duljinaPodniz)
	{
		pronadjeno = 1;
		for(j=0;j<duljinaPodniz;j++)
		{
			if (niz[i+j] != podniz[j])
			{
				pronadjeno = 0;
				break;
			}
		}
		if (pronadjeno == 1)
		{
				for(j=i;j<=duljinaNiz-duljinaPodniz;j++)
					niz[j] = niz[j+duljinaPodniz];
				duljinaNiz -= duljinaPodniz;
		}
		else
			i++;

	}

	puts(niz);
	return 0;
}
```

10. Zadatak

Učitajte dvije matrice, na način da prvo učitate m i n (broj redaka i broj stupaca prve), a potom n i k (broj redaka i broj stupaca druge). Moguće je pretpostaviti da matrice neće biti već od dimenzija 25 x 25. Za definiranje dimenzija koristite simboličke konstante. Ukoliko nisu učitane ispravne vrijednosti (matrice nisu ulančane, broj je manji ili jednak 0 ili veći od 25), učitajte sve tražene podatke ponovo.

Potom u treću matricu matrično izmnožite prvu sa drugom (A * B) te ispišite na ekran.

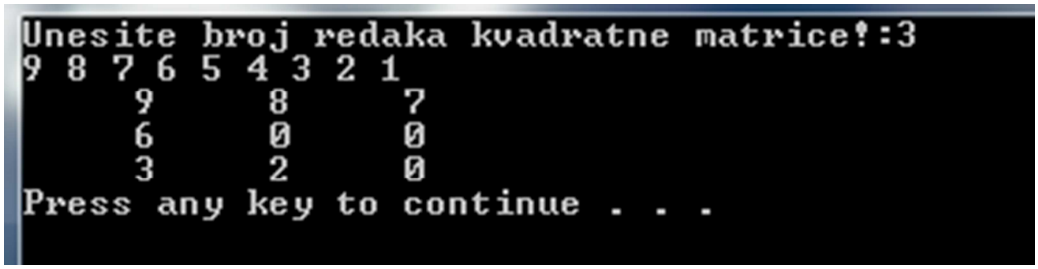
```c
#include <stdio.h>
#define MAX 25

int main ()
{
	int matricaA[MAX][MAX];
	int matricaB[MAX][MAX];
	int matricaC[MAX][MAX] = {0};
	int brredA, brredB, brstupA, brstupB;
	int n,i,j,k;

	do
	{
		scanf ("%d %d %d %d", &brredA, &brstupA, &brredB, &brstupB);
	}while (brredA <= 0 || brredB <= 0 || brstupA <= 0 || brstupB <= 0 || brredA > MAX ||
		brredB > MAX || brstupA > MAX || brstupB > MAX || brstupA != brredB);

	for (i=0;i<brredA;i++)
		for (j=0;j<brstupA;j++)
			scanf ("%d", &matricaA[i][j]);

	for(i=0;i<brredB;i++)
		for (j=0;j<brstupB;j++)
			scanf ("%d", &matricaB[i][j]);

	for(i=0;i<brredA;i++)
		for(j=0;j<brstupB;j++)
			for (k=0;k<brstupA;k++)
			{
				matricaC[i][j] += matricaA[i][k] * matricaB[k][j];
			}

	for(i=0;i<brredA;i++)
	{
		for(j=0;j<brstupB;j++)
			printf ("%6d", matricaC[i][j]);
		printf ("\n");
	}

	return 0;
}
```

Greška: Bilo je više grešaka koje su ispravljene (rezultiralo krivim množenjem i krivim ispisom). (Velik broj različitih dimenzija je pomalo zbunjujuć, isplati se program korak po korak izvest za matrice recimo (3,3) x (3,4) ili (3,1) x (1,3) te (1,3) x (3,1).

11. Zadatak

Učitati broj redaka kvadratne matrice, a potom i elemente same matrice. Može se pretpostaviti da matrica neće biti dimenzija većih od 10 x 10. Potom sve elemente iznad glavne dijagonale (uključivo i dijagonala) koji su strog manji od najvećeg elementa ispod sporedne dijagonale (uključivo) postavite na nulu. Takvu matricu ispisati na ekran. Za definiranje najvećih dimenzija matrice obavezno koristite simboličke konstante. Također, obavezno provjerite jeli učitana stvarna dimenzija matrice ispravna (korisnik ne smije unijeti 0, negativan broj ili više od 10). Korisniku nuditi unos dimenzije dok god ne učita ispravan podatak.

Primjer:



Rješenje:

```c
#define MAX 10
#include <stdio.h>

int main ()
{
	int matrica [MAX][MAX];
	int n,i,j, max;
	do {
		printf ("Unesite broj redaka kvadratne matrice!:");
		scanf ("%d", &n);
	} while (n<=0 || n> MAX);

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

	max = matrica[0][n-1];
	for (i=0;i<n;i++)
		for (j=n-i-1;j<n;j++)
			if (max < matrica[i][j])
				max = matrica[i][j];

	for (i=0;i<n;i++)
		for (j=i;j<n;j++)
			if (matrica[i][j] < max)
				matrica[i][j] = 0;
	for (i=0;i<n;i++)
	{
		for (j=0;j<n;j++)
			printf ("%6d", matrica[i][j]);
		printf ("\n");
	}

	return 0;
}
```

Napomena: Na instrukcijama je bilo predloženo rješenje da se prolazi kroz čitavu matricu, ali da se primjerice za glavnu dijagonalu (i iznad njeg) mijenjaju brojevi samo tamo gdje je `i <= j`. To je dobro

rješenje, ali neučinkovito. Primjerice, pretpostavimo postojanje matrice 1000 x 1000 (ne u ovom zadatku, ali u nekom drugom ozbiljnom zadatku). Na taj način ćemo prošetati kroz svih 1000 x 1000 elemanta matrici i 1000 x 1000 obaviti provjeru jeli i <= j. Što je vrlo neučinkovito. Ovdje napisanim rješenjem automatski određujemo kroz koja polja zaista prolazimo. To je učinkovitije rješenje.

Također je ispravljena jedna greška (pisalo je na početku `n >= MAX`, treba stajati `n > MAX`).

12. Zadatak

Unesite kvadratnu matricu, dimenzija ne većih od 20 x 20. Unesite i realne dimenzije matrice (nije potrebno provjeravati ispravnost). Potom unesite n. N označava koliko stupaca se matrica mora pomaknuti ulijevo. Pritom krajnje lijevi stupac ulazi na mjesto krajnje desnog!

```
#define MAX 20
#include <stdio.h>

int main ()
{
	int matrica[MAX][MAX];
	int n, i, j,k, pom;
	int pomak;

	scanf ("%d", &n);

	for (i=0;i<n;i++)
		for (j=0;j<n;j++)
			scanf ("%d", &matrica[i][j]);

	scanf ("%d", &pomak);

	pomak = pomak %n;

	for (k=0;k<pomak;k++)
	{
		for (i=0;i<n;i++)
		{
		pom = matrica[i][0];
		for (j=0;j<n-1;j++)
		{
			matrica[i][j] = matrica[i][j+1];
		}
		matrica[i][n-1] = pom;
		}
	}

	for (i=0;i<n;i++)
	{
		for (j=0;j<n;j++)
			printf ("%6d", matrica[i][j]);
		printf ("\n");
	}

	return 0;
}
```

13.Zadatak

Za matricu iz prethodnog zadatka učinite sljedeće. Također unesite dva intervala. Na ekran ispišite sve brojeve koji su bili djeljivi sa barem jednim brojem iz prvog intervala, sa barem jednim brojem iz drugog intervala, te sa barem jednim brojem iz oba intervala (djelitetlji ne moraju biti isti!). Međutim, navedenu provjeru vršite samo za elemente koji čine pješčani sat kada se povuku sporedna i glavna dijagonala na matrici. Možete koristiti pomoćna polja za pohranu podataka. Može se pretpostaviti da su u matricu unošeni jedinstveni brojevi (nema ponavljanja nekog broja).

```
#define MAX 20
#include <stdio.h>

int main ()
{
	int matrica[MAX][MAX];
	int prviInterval[MAX], drugiInterval[MAX];
	int brojacPrvi, brojacDrugi;
	int n, i, j,k, pom;
	int pomak;
	int dA,gA,dB,gB;

	scanf ("%d", &n);

	for (i=0;i<n;i++)
		for (j=0;j<n;j++)
			scanf ("%d", &matrica[i][j]);

	scanf ("%d", &pomak);

	pomak = pomak %n;

	for (k=0;k<pomak;k++)
	{
		for (i=0;i<n;i++)
		{
		pom = matrica[i][0];
		for (j=0;j<n-1;j++)
		{
			matrica[i][j] = matrica[i][j+1];
		}
		matrica[i][n-1] = pom;
		}
	}

	for (i=0;i<n;i++)
	{
		for (j=0;j<n;j++)
			printf ("%6d", matrica[i][j]);
		printf ("\n");
	}

	printf ("Intervali!");
	scanf ("%d %d %d %d", &dA, &gA, &dB, &gB);
```

```
brojacPrvi = 0;
	brojacDrugi = 0;
	for(i=0;i<n/2;i++)
		for (j=i;j<=n-1-i;j++)
		{
			for (k = dA;k<=gA;k++)
				if (matrica[i][j] % k == 0)
				{
					prviInterval[brojacPrvi] = matrica[i][j];
					brojacPrvi++;
					break;
				}
			for (k = dB;k<=gB;k++)
				if (matrica[i][j] % k == 0)
				{
					drugiInterval[brojacDrugi] = matrica[i][j];
					brojacDrugi++;
					break;
				}
		}


	for (i=n/2;i<n;i++)
		for (j=n-1-i;j<=i;j++)
		{
			for (k = dA;k<=gA;k++)
				if (matrica[i][j] % k == 0)
				{
					prviInterval[brojacPrvi] = matrica[i][j];
					brojacPrvi++;
					break;
				}
			for (k = dB;k<=gB;k++)
				if (matrica[i][j] % k == 0)
				{
					drugiInterval[brojacDrugi] = matrica[i][j];
					brojacDrugi++;
					break;
				}
		}

	for (i=0;i<brojacPrvi;i++)
		printf ("%d ", prviInterval[i]);
	printf ("\n");
	for (i=0;i<brojacDrugi;i++)
		printf ("%d ", drugiInterval[i]);
	printf ("\n");

	for (i=0;i<brojacPrvi;i++)
		for (j=0;j<brojacDrugi;j++)
			if (prviInterval[i] == drugiInterval[j])
				printf ("%d ", prviInterval[i]);
	printf ("\n");
	return 0;
}
```