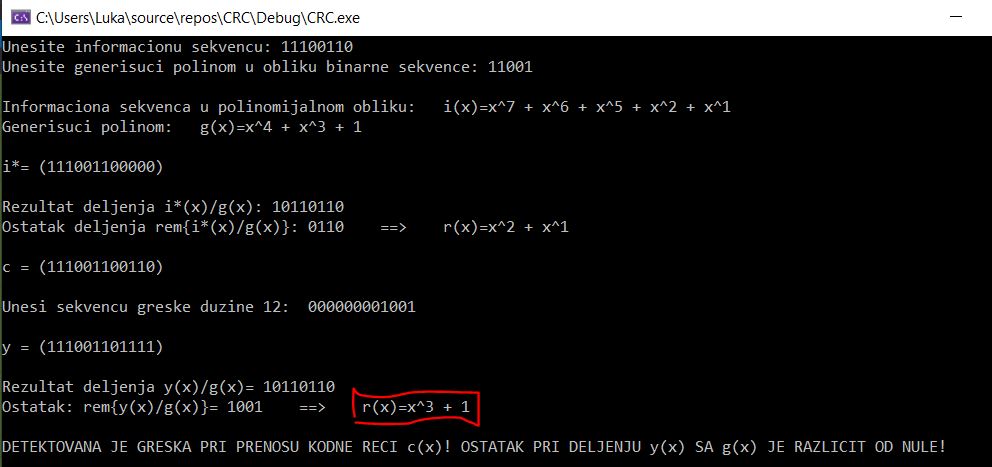
OTR - Domaći zadatak

**Cyclic redundancy check (CRC)**

**\*NAPOMENA:** Za razliku od materijala sa časa, ovde u binarnoj sekvenci „najviši“ biti predstavljaju koeficijente **većeg** stepena pa tako npr. sekvenca 11001 odgovara polinomu: x4 + x3 + 1, dok bi takav polinom na materijalima bio predstavljen kao sekvenca 10011. To dovodi do određenih promena, npr. kada se informaciona sekvenca proširuje određenim brojem 0, ti biti se dodaju na kraj sekvence, a ne na početak, pa se i ostatak pri deljenju i\*(x) sa g(x) XOR-uje sa tim poslednjim bitima, a ne sa prvim itd.

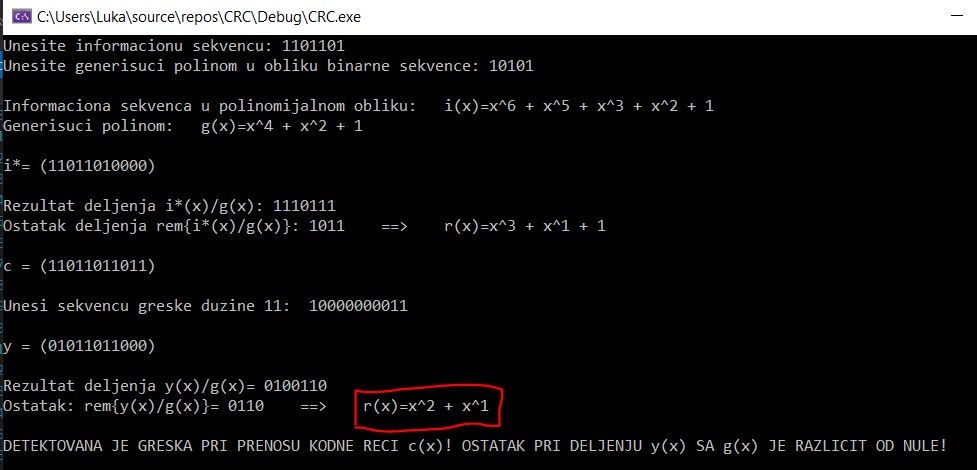
U nastavku su data 3 primera.

Primer 1 (Primer sa prezentacije, slajdovi 27 i 28):

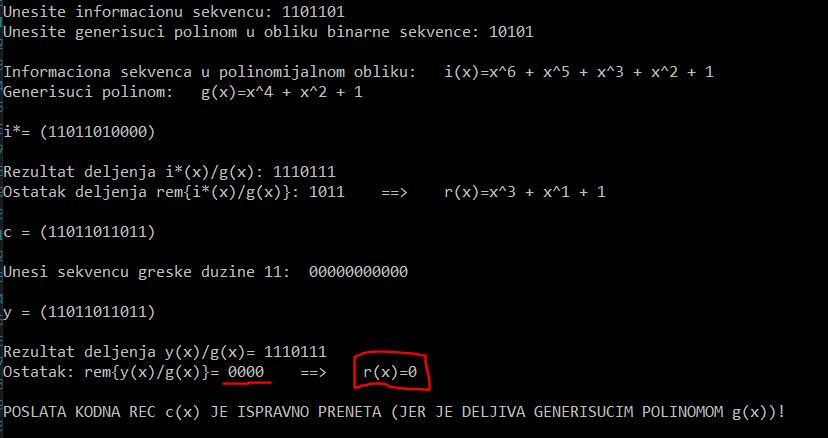


Primer 2:

1. Neispravno preneta sekvenca (detektovana greška)

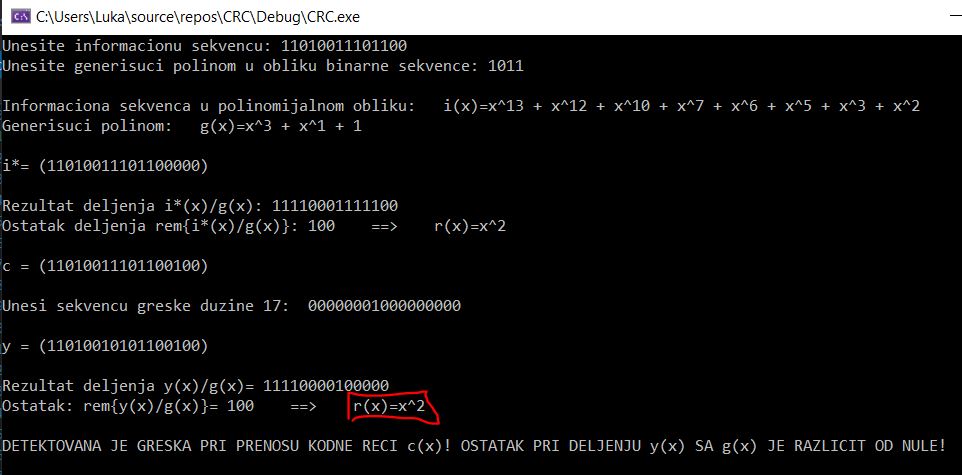


1. Ispravno preneta sekvenca

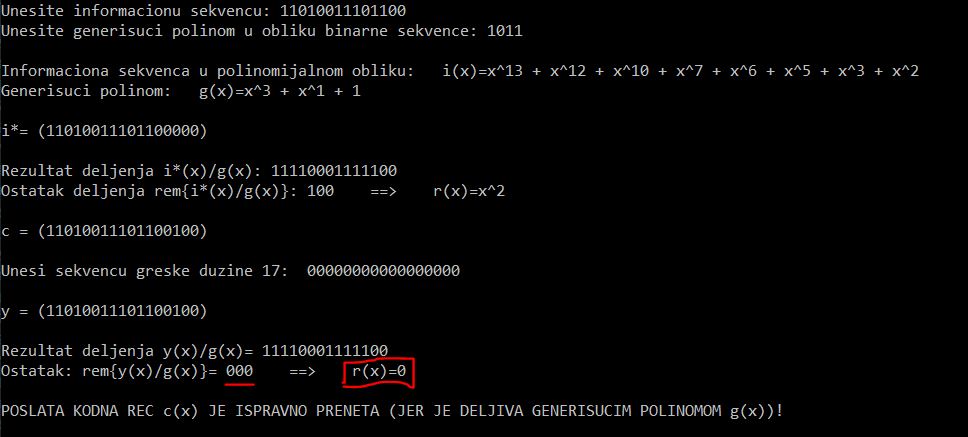


Primer 3:

1. Neispravno preneta sekvenca (detektovana greška)



1. Ispravno preneta sekvenca



**Komentar**:

Pored toga što se veoma jednostavno implementira hardverski, ovaj algoritam se i jednostavno analizira matematički jer se svakoj binarnoj sekvenci može pridružiti polinom. Ovakav kod ne zahteva 2D predstavu (generišuću matricu G), već je dovoljna i 1D predstava (generišući polinom g(x)). Ipak, mana CRC-a je da on služi samo za detektovanje grešaka i ne može da ih ispravi. Ne može da se zna na kojoj poziciji se nalazi greška, kao ni da li je pri prenosu postojala jedna ili više grešaka.

Što se tiče programa, korišćene su iste oznake kao na prezentaciji. Sa i(x) je označena informaciona sekvenca (i odgovarajući polinom), sa g(x) generišući polinom, i\* predstavlja infromacionu sekvencu nakon proširenja sa određenim brojem 0, r(x) i rem su oznake za ostatke pri deljenju, c(x) je kodna reč, e je sekvenca greške, dok je y sekvenca koja je stigla na stranu receiver-a.

Pored binarnih sekvenci, ostaci pri deljenju su prikazani i kao polinomi, a tako su predstavljeni i infomraciona sekvenca i generišući polinom nakon unosa istih.

Program sadrži i odgovarajuće provere pri unosu, pa se tako proverava da li je za i(x), g(x) i e(x) uneta isključivo binarna sekvenca (ako se unese bilo koji znak osim 0 i 1, traži se ponovni upis). Takođe, zahteva se i da dužina sekvence greške bude dužine i kao kodna reč (uvek piše kolika je tačno to sekvenca).

Na samom kraju programa, ispisuje se rezultat (tj. poruka), a to je da li je detektovana greška ili je kodna reč ispravno preneta. To, naravno, zavisi od ostatka pri deljenju y(x) sa g(x): Ako je ostatak 0, reč je ispravno preneta, u suprotnom nije.

Potom se korisnik pita da li želi da završi sa korišćenjem programa ili hoće još da ga koristi pa se očekuje da unese 0 ako ne želi da završi ili 1 ako želi.

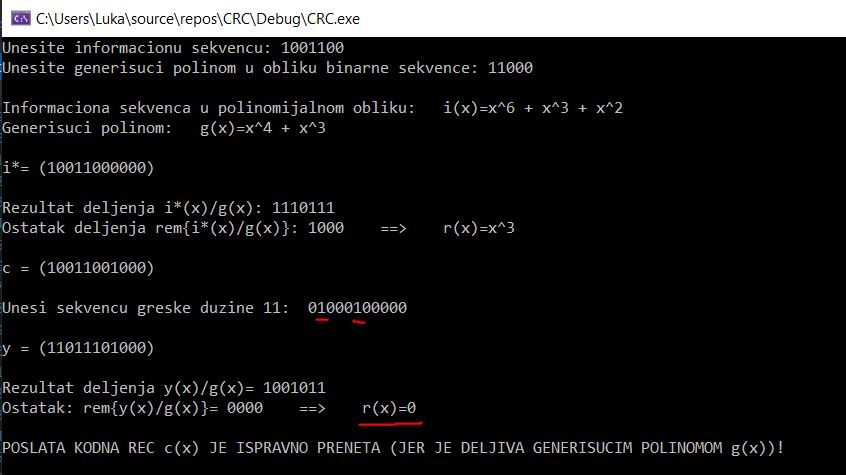
**CRC metod ume da pogreši!**

Ne može se svaki put detektovati greška pri prenosu iako na prvi pogled ovaj kod se čini savršenim. Naime, kada ostatak pri deljenju sekvenci i\*/g dodamo na i\*, mi dobijamo kodnu reč c koja je sigurno deljiva generišućim polinomom. Onda, na tu sekvencu dodajemo sekvencu greške i ako je dobijena sekvenca y deljiva generišućim polinomom, smatramo da je reč ispravno preneta. Ako je sekvenca greške e samo niz 0, tj. e=(000...0), kodna reč c i reč koja stiže na prijemnik y su iste sekvence, pa je i logično da generišući polinom deli y (jer je delio i c). Generišući polinom će deliti y onda kada deli i kodnu reč (uvek) i sekvencu greške (a sekvenca 000...0 je svakako jedna od tih), ali problem nastaje jer sekvenca koja je niz 0 ne mora da bude jedina sekvenca greške za koju će da važi rem{e(x)/g(x)}=0.

Zbog toga, izazov CRC koda je taj minimizira broj grešaka zbog kojih je rem{e(x)/g(x)}=0, a jedno od rešenja je da se generiše što bolji polinom g(x) takav da on deli što manje sekvenci.

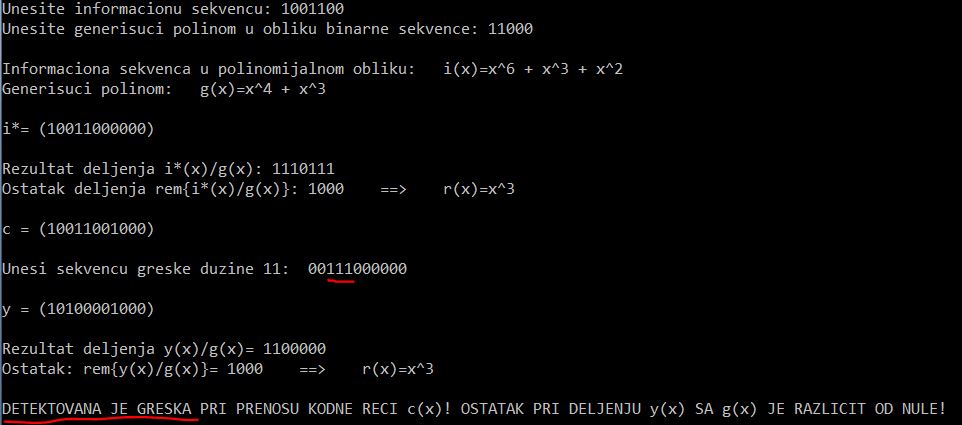
Sledi par primera na kojim ćemo pokazati kako i zašto CRC ume da pogreši.

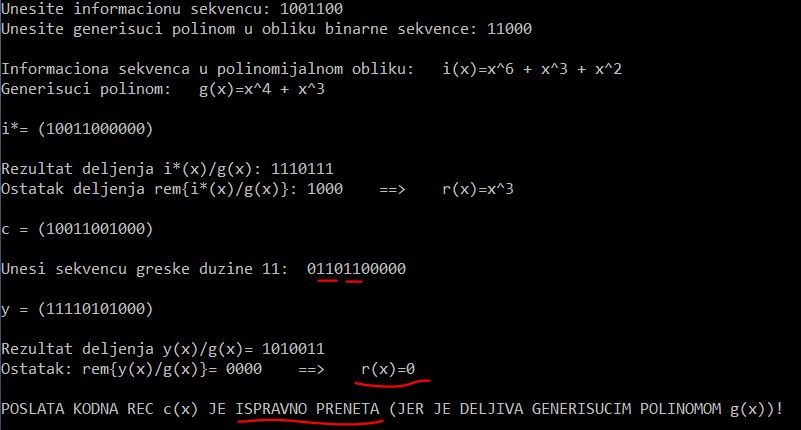
Primer 4 (Sekvenca greške nije detektovana):



Kao što vidite, ostatak pri deljenju y/g je 0 pa dobijamo poruku da je kodna reč ispravno preneta iako je postojala greška u prenosu, tj. sekvenca greške nije detektovana. To je zato što sama sekvenca greška je deljiva generišućim polinomom, tj. g deli e.

Najmanji broj jedinica u sekvenci greške koji će dovesti do ovakve pojave je broj jedinica u samom generišućem polinomu (u ovom primeru to su 2 jedinice), ali to ne znači da će svaki veći broj jedinica dovesti do ovakve pojave. Potrebno je da taj broj jedinica bude deljiv brojem jedinica u generišućem polinomu. Pa tako, u datom primeru, ukoliko bi sekvenca greške imala 2,4,6,.. jedinica, greška ne bi bila detektovana. U slučaju da je taj broj jedinica jednak 3,5,... CRC će detektovati grešku u prenosu. Slede primeri u kojima ćemo za isti gen. polinom videti rezultate ukoliko sekvenca greške ima 3 i ukoliko ima 4 jedinice.

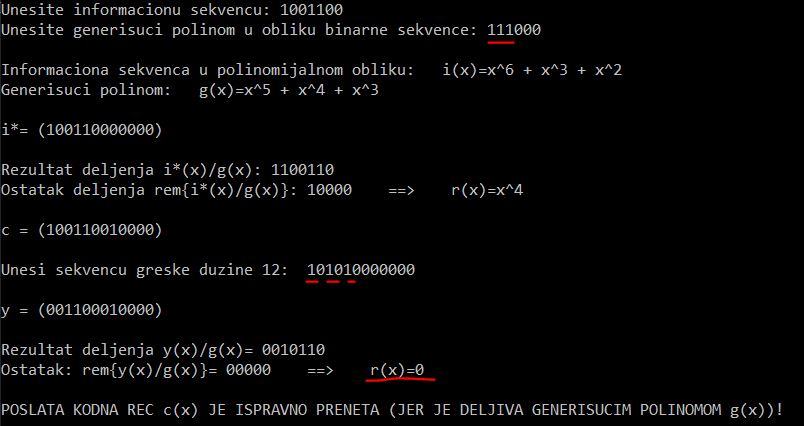




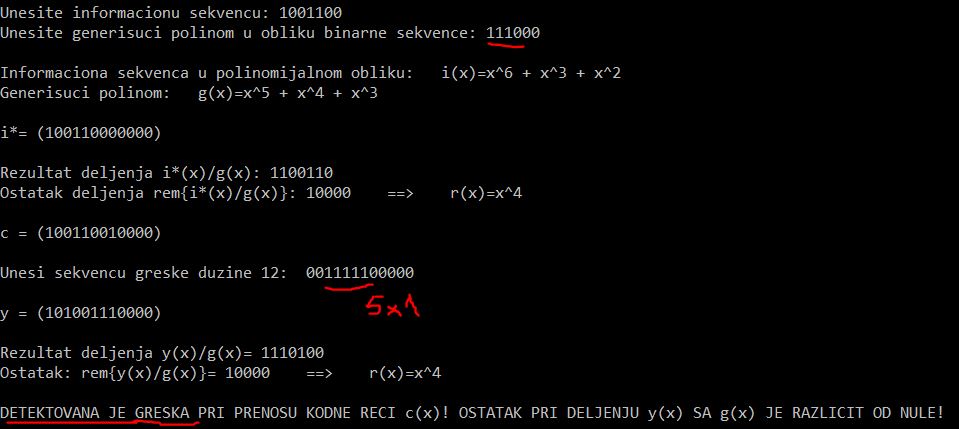
Primer 5:

Evo još jedan primer u kome ćemo videti kako greška nije detektovana iako postoji. U ovom primeru, generišući polinom ima 3 jedinice. Ako sekvenca greške ima 3 jedinice (isti broj 1 kao u gen. polinomu), ona neće biti detektovana i dobićemo poruku da je kodna reč ispravno preneta. Istu poruku ćemo dobiti i ako sekvenca ima greške ima 6 jedinica (jer je 6 deljivo sa 3), dok u slučaju da npr. imamo 4 ili 5 jedinica u sekvenci greške, greška će biti detektovana.

3 jedinice u sekvenci greške => greška nije detektovana



5 jedinica u sekvenci greške => greška je detektovana



6 jedinica u sekvenci greške => greška nije detektovana

