ZADACI IZ C-PROGRAMIRANJA

Zadatak koji slijedi primjer je "programerskog sljepila". A priča uz njega vezana ide ovako:
U vrijeme dok sam sređivao biblioteku libbsutl funkcija za komunikaciju s bazom podataka (PostgreSQL) počeli su se pojavljivati segmentation faultovi pri pozivu dolje navedene funkcije i to pri izvršavanju linije označene strelicom (PQclear(resgls[rn]);) ali ne uvijek, nego u nekim okolnostima. Onda sam komentirao liniju PQclear(resglt); (kako se vidi iz priloženog) i izgledalo je da je sve OK. Nakon dan-dva opet su se počeli pojavljivati segmentation faultovi na istoj liniji pa sam par sati kopao po programu tražeći gdje bi mogla biti greška (mislio sam da se radi o nekoj krivoj alokaciji/owerflowu ili sl.). A rješenje je više nego očigledno...
(Da pojasnim - ovdje se radi o brisanju rn recordsetova koji su povučeni iz baze radi kreiranja reporta i strpani u array PGresult *resgls[ ])

```
/* Brisanje recordsetova za report */

void clear_reprecset(int rn)
{
  int i;
  /* PQclear(resglt);*/
  /* zadnji resgls[] je resglt pa imamo segmentation fault pri ovakvom PQclear() */
  for (i=0; i<rn; i++)
  {
--> PQclear(resgls[rn]);
  }
}
```

Evo još jedne funkcije "s greškom". Ako se dolje navedena funkcija pozove, program će se srušiti na liniji označenoj strelicom nakon većeg broja poziva ove funkcije (dakle ne odmah pri prvom pozivu). Treba pronaći u čemu je problem.
Inače, nije teško vidjeti da navedena funkcija služi za spajanje dvaju stringova u (ch)separated (dvočlani) niz (osim u slučaju ch = 0 kad se stringovi lijepe jedan za drugi)

```
char *concatens(char *strn, char *strm, char ch)
{
  int i, j, l;
  int jm = 1;
  int ln = 0;
  int lt[2];
  char *tstr[2];
  char *cstr;

  if (strn == NULL && strm == NULL)
    return NULL;
  else if (strn == NULL)
    tstr[0] = strm;
  else if (strm == NULL)
    tstr[0] = strn;
  else {
    jm = 2;
    tstr[0] = strn;
```

```
    tstr[1] = strm;
    lt[1] = strlen(tstr[1]);
  }

  lt[0] = strlen(tstr[0]);
  l = ((ch) != (0)) ?  (jm + 1) :  (1);
  cstr = malloc(lt[0] + lt[1] + l);

  for (j=0; j<jm; j++) {
    for(i=0; i<lt[j]; i++)
-->   *(cstr+ln+i) = *(tstr[j]+i);
    if (ch != 0) {
      *(cstr+lt[j]+ln) = ch;
      ln = ln + lt[j] + 1;
    }
    else
      ln = ln + lt[j];
  }
  *(cstr+ln) = 0;
  return cstr;
}
```

Pretpostavimo da je korištenjem funkcije char *concatens() iz prethodnog zadatka (dakako one kod koje je ispravljena greška koja dovodi do segmentation faulta) potrebno spojiti n stringova, tako da su neki od njih "slijepljeni" (%s%s), dok su neki odvojeni spaceom (%s %s), na sve moguće načine.
Koliko je najmanje puta potrebno primijeniti ovu funkciju, da bi se dobila neka od tih kombinacija?
Odredi sve takve kombinacije (kombinacije lijepljenja i odvajanja spaceom n stringova koje se mogu dobiti primjenom funkcije char *concatens() minimalni broj puta).
Odredi takodjer maksimalni broj primjena funkcije char *concatens(), pomoću kojih se može dobiti neka od opisanih kombinacija n stringova, kao i sve takve kombinacije (kombinacije lijepljenja i odvajanja spaceom n stringova koje se mogu dobiti primjenom funkcije char *concatens() maksimalni broj puta).
Da li se za svaki broj primjena funkcije char *concatens() između ranije utvrđenih maksimalnog i minimalnog može dobiti neka od opisanih kombinacija stringova?

Napiši C program s minimalnim brojem linija koji daje sva rješenja diofantske jednadžbe
$2n(n+1) = m(m+1), \quad m < k$
gdje su k, m, n prirodni brojevi. k neka bude argument programa koji se zadaje pri njegovom startanju.

Kakve sve promjene treba napraviti na (Linux) sistemu, da bi bash komanda whoami ispisivala poruku "You are genious" u slučaju kad je korisnik dobre, odnosno poruku "You are idiot" u slučaju kad je loše volje?