1 Veštačka inteligencija 2015/16, test 2, 05. april 2016.

Zadatak 1 *(**minimax**) Raša i Pera igraju uprošćenu verziju igre uno. Na raspolaganju imaju karte:* `0b, 1c, 1b, 4b, 1b, 2c, 2b, 3b` *(svaka karta je označena brojem i bojom,* `c` *označava crnu, a* `b` *označava belu boju). Na početku svaki od igrača dobije po četiri karte. Pobednik je onaj igrač koji prvi utroši sve karte. Pravila igre su sledeći:*

- *Prvi igrač može da stavi bilo koju kartu.*
- *Igrač može staviti kartu iste boje kao poslednja odigrana karta ili kartu koja ima isti broj kao poslednja odigrana karta.*
- *Ako igrač ne može da stavi ni jednu kartu, on ne igra, sledeći potez igra drugi igrač.*

Raša zeli da zna kako u svakom trenutku najbolje da igra, pa je odlučio da napiše program koji će da mu pomogne. Kao algoritam koristi minimax. Ipak, Raša nije tako dobar programer i ne zna da napiše kompletan algoritam. Raša je za sada napisao:

```
typedef struct
{
   int boja; //moze biti 1(crna) ili 0(bela)
   int broj;
   int odigrana; //oznacava da li je karta odigrana, 0 ako jeste, 1 ako nije
}_karta;


// rasa[4] promenljiva koja oznacava karte koje ima rasa
// pera[4] promenljiva koja oznacava karte koje ima pera
//funkcija koja ispituje da li se stiglo do kraja, vraca 1 ako je kraj igre, u suprotnom nula
//u promenljivu rez upisuje:
//1 ako je pobedio rasa
//-1 ako je pobedio pera
//0 ako je nereseno
//
int kraj(int* rez, _karta rasa[4], _karta pera[4]);

//potez je karta koji treba odigrati; ukoliko nije moguce odigrati postavlja se na NULL
//poslednje_odigrana -- karta koja je poslednja odigrana i nalazi se na stolu
//ukoliko nema karata na stolu, onda su sve vrednosti u poslednja_odigrana postavljene na -1
//
int max(_karta* potez, _karta rasa[4], _karta pera[4], _karta poslednje_odigrana)
{
   int rez;
   int maxv=-2;

   if (kraj(&rez, rasa, pera))
       return rez;

   ...

}

void minimax(_karta* potez, _karta rasa[4], _karta pera[4], _karta poslednje_odigrana)
{
    max(potez, rasa, pera, poslednje_odigrana);
}
```

Pomozite Raši da dovrši funkciju `max` *(pretpostavljamo da će sam znati da napiše funkciju* `min` *i potrebne korake u* `main-u`*).*

Zadatak 2 ***A****

a) *Algoritam A* se ne zaustavlja sve dok se ciljni čvor ne izbaci iz otvorene liste. To znači da algoritam može imati i milion iteracija pre nego što izbaci čvor iz otvorene liste, što može delovati neefikasno. Zašto ne zaustaviti A* čim se ciljni čvor doda u otvorenu listu.*

---

b) *Konstruisati primer koji pokazuje zašto je važno algoritam zaustaviti tek nakon izbacivanja ciljnog čvora iz otvorene liste. Obrazložiti navedeni primer. (U primeru ne koristiti više od četiri čvora).*

Zadatak 3 ***(genetski algoritmi)*** *Genetskim algoritmom se traži maksimum funkcije* $-5 \cdot x^2 + 8 \cdot x + 20$ *. Populacija sadrži (samo) jedinke (1), (2), (0) i (-1). Za svaku od jedinki, izračunati verovatnoću da će biti izabrana za reprodukciju u ruletskoj selekciji.*