Strukture podataka i algoritmi 2
Septembar, 2012

1. Napisati funkciju za sortiranje povezane liste celih brojeva korišćenjem Quick Sort algoritma, bez korišćenja pomoćnih nizova.

   (30 poena)

U nastavku su data neka od rešenja studenata.

Rešenje 1
(Miloš Furtula)

```
#include <stdio.h>
#include <stdlib.h>

struct clan
{
	int broj;
	struct clan *sledeci;
};

struct clan *poc,*kraj;

#define novi(x) x=(struct clan*)malloc(sizeof(struct clan))

void quick_sort(struct clan *donja,struct clan *gornja)
{
	struct clan *pivot,*bef,*aft,*i;
	int k;

	if ((donja==gornja)||(donja==NULL)||(gornja==NULL)) return;
	if ((donja->sledeci==gornja)&&(donja->broj>gornja->broj))
	{
		k=gornja->broj;
		gornja->broj=donja->broj;
		donja->broj=k;
	}

	bef=donja;
	pivot=donja;
	aft=pivot->sledeci;
	i=donja->sledeci;

	while (i!=gornja->sledeci)
	{
		if (i->broj<donja->broj)
		{
			bef=pivot;
			pivot=pivot->sledeci;
			aft=pivot->sledeci;
			k=pivot->broj;
			pivot->broj=i->broj;
			i->broj=k;
		}
		i=i->sledeci;
	}

	k=pivot->broj;
	pivot->broj=donja->broj;
	donja->broj=k;

	if (aft==gornja->sledeci) aft=NULL;
```

```
	quick_sort(donja,bef);
	quick_sort(aft,gornja);
}

main()
{
	int i,n;
	struct clan *tek,*temp;
	scanf("%d",&n);
	novi(poc);
	scanf("%d",&poc->broj);
	poc->sledeci=NULL;
	tek=poc;
	for (i=2;i<=n;i++)
	{
		novi(temp);
		tek->sledeci=temp;
		scanf("%d",&temp->broj);
		temp->sledeci=NULL;
		tek=temp;
	}
	kraj=temp;
	printf("\n");
	quick_sort(poc,kraj);
	temp=poc;
	for (i=1;i<=n;i++)
	{
		printf("%d ",temp->broj);
		temp=temp->sledeci;
	}
	printf("\n");
}
```

Rešenje 2
(Nataša Antić 54/09)

```
#include<stdio.h>
#include<stdlib.h>
typedef struct lista
{
	int podatak;
	struct lista *sledeci;
} LISTA;

LISTA* dodaj(LISTA *l,int t);
void QuickSort(LISTA **p, LISTA *kraj);
void dodajNaPocetak(LISTA **q,LISTA **p,LISTA **tmp);
void ispis_liste(LISTA *p);

LISTA* dodaj(LISTA *l,int t)
{
	LISTA *p=l,*element;
	element=(LISTA *)malloc(sizeof(LISTA));
	element->podatak=t;
	element->sledeci=NULL;

	if(l==NULL)
		return element;
	while(p->sledeci!=NULL) p=p->sledeci;
	p->sledeci=element;
	return l;
}
```

```
void QuickSort(LISTA **p, LISTA *kraj)
{
	if((*p)==NULL) return;

	LISTA *pivot=*p;
	LISTA *q=*p;
	LISTA *tmp=q->sledeci;

	if(q->sledeci!=NULL && tmp->podatak<pivot->podatak)
		dodajNaPocetak(&q,p,&tmp);
	while(q->sledeci!=kraj)
	{
		if(tmp->podatak<pivot->podatak)
			dodajNaPocetak(&q,p,&tmp);
		else
		{
			tmp=tmp->sledeci;
			q=q->sledeci;
		}
	}

	LISTA *krajprve=pivot;
	if((*p)!=krajprve)
		QuickSort(p,krajprve);
	pivot=krajprve->sledeci;
	if(pivot!=kraj)
		QuickSort(&pivot,kraj);

	krajprve->sledeci=pivot;
}

void dodajNaPocetak(LISTA **q,LISTA **p,LISTA **tmp)
{
	(*q)->sledeci=(*q)->sledeci->sledeci;
	(*tmp)->sledeci=(*p);
	(*p)=(*tmp);
	(*tmp)=(*q)->sledeci;
}

void ispis_liste(LISTA *p)
{
	if(p==NULL)
	{
		printf("Lista je prazna.\n");
		return;
	}
	while(p!=NULL)
	{
		printf("%d ",p->podatak);
		p=p->sledeci;
	}
	printf("\n");
}

main(int argc,char *argv[])
{
	LISTA *p=NULL;
	int brojElemenata,element,i;
	printf("Unesite broj elemenata liste:");
	scanf("%d",&brojElemenata);
	for(i=0;i<brojElemenata;i++)
	{
		scanf("%d",&element);
		p=dodaj(p,element);
	}
	printf("Lista PRE sortiranja:\n");
	ispis_liste(p);
```

```
        QuickSort(&p,NULL);
        printf("Lista POSLE sortiranja:\n");
        ispis_liste(p);
}
```