

Rešitve pisnega izpita z dne 17. junija 2003

1. naloga (25%)

```
void trikotnik(int n) {
    int i, j;

    for( i=0; i<(n+1); i++ ) {
        for( j=n-i; j>0; j-- )
            printf(" ");
        for( j=0; j<(2*i+1); j++ )
            printf("*");
        printf("\n");
    }
}
```

2. naloga (25%)

```
int drugi(int polje[], int vel) {
    int i, naj1, naj2;

    if( vel < 2 )
        return(-1);
    if( polje[0] < polje[1] ) {
        naj1 = polje[0];
        naj2 = polje[1];
    }
    else {
        naj1 = polje[1];
        naj2 = polje[0];
    }
    for( i=2; i<vel; i++ ) {
        if( polje[i] <= naj1 ) {
            naj2=naj1;
            naj1=polje[i];
        } else
            if( polje[i] <= naj2 ) {
                naj2=polje[i];
            }
    }
    return(naj2);
}
```

ali na drug način

```
int drugi(int polje[], int vel) {
    int i,j,tmp;

    if( vel < 2 )
        return(-1);
    for( i=0; i<2; i++ )
        for( j=i+1; j<vel; j++ )
            if( polje[j] < polje[i] ) {
                tmp=polje[j];
                polje[j]=polje[i];
                polje[i]=tmp;
            }
    return(polje[1]);
}
```

3. naloga (25%)

```
#include <stdio.h>
#define MAX_CRK 26

main(int argc, char *argv[]) {
    FILE *fp;
    int ch;
    int crke[MAX_CRK];

    for(ch=0; ch<MAX_CRK; ch++)
        crke[ch] = 0;
    fp = fopen(argv[1], "r");
    while( (ch=getc(fp)) != EOF ) {
        if( (ch >= 'a') && (ch <= 'z') )
            ++crke[ch-'a'];
        if( (ch >= 'A') && (ch <= 'Z') )
            ++crke[ch-'A'];
    }
    fclose(fp);
    for(ch='a'; ch<='z'; ch++)
        printf("Crka %c se pojavi %d-krat.\n", ch, crke[ch-'a']);
}
```

4. naloga (25%)

```
struct element {
    int prioriteta;
    struct element *naslednji;
};

// predpostavimo, da so elementi v vrsti urejeni NARASCAJOCE
// (manjše stevilo pomeni visjo prioriteto)

void dodaj(struct element **p, int e) {
    struct element *q, *r;

    q = (struct element*) malloc(sizeof(struct element));
    q->prioriteta = e;
    if ( (*p == NULL) || (e <= (*p)->prioriteta) ) { // dodaj na zacetek seznama
        q->naslednji = *p;
        *p = q;
    } else {
        // poisci ustrezno mesto v seznamu za nov element
        r = *p;
        while ( (r->naslednji != NULL) && (r->naslednji->prioriteta < e) )
            r = r->naslednji;
        // dodaj za element, na katerega kaze r
        q->naslednji = r->naslednji;
        r->naslednji = q;
    }
}
```