

1. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

void fun(void) {
    int x = 5, y = 15;
    {
        int x = 20;
        y = 25;
        printf("x = %d, y = %d\n", x, y);
    }
    printf("x = %d, y = %d\n", x, y);
}

int main(void) {
    int x = 100;
    printf("x = %d\n", x);
    fun();
    fun();
    printf("x = %d\n", x);
    return 0;
}
```

2. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

static int x = 10;

void fun(void) {
    static int x = 5;
    printf("x = %d\n", x);
    x += 3;
}

int main(void) {
    fun();
    printf("x = %d\n", ++x);
    fun();
    printf("x = %d\n", ++x);
    return 0;
}
```

3. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

void fun(void) {
    static int x = 5;
    int y = 5;
    printf("%d %d\n", ++x, --y);
}

int main(void) {
    fun();
    fun();
    fun();
    return 0;
}
```

4. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

int main(void) {
    static int i = 5;
    int prviPut = 1;
    labela:
    {
        static int i = 10;
        int j = 15;
        printf("%d %d\n", i, j);
        i++;
        j++;
    }
    i++;
    printf("%d\n", i);
    if (prviPut) {
        prviPut = 0;
        goto labela;
    }

    return 0;
}
```

5. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

static int x = 25;

void fun1(void) {
    static int x = 5;
    printf("%d\n", ++x);
}

void fun2(void) {
    int x = 10;
    printf("%d\n", ++x);
}

void fun3(void) {
    printf("%d\n", ++x);
}

int main(void) {
    x++;
    {
        static int x = 15;
        {
            int x = 20;
            printf("%d\n", x++);
        }
        printf("%d\n", x++);
    }
    printf("%d\n", x++);
    fun1();
    fun2();
    fun3();

    fun1();
    fun2();
    fun3();

    return 0;
}
```

6. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

#define MAXNIZ 10
#define MAXTOC 5

typedef struct {double x; double y;} tocka_t;
static tocka_t tocke[MAXTOC] = {{1., 1.}, {2., 2.}};
static tocka_t *pTocka = &tocke[0];

static char niz[10 + 1];

void f1(void) {
    char niz[MAXNIZ + 1] = "Ivan";
    printf("%s!\n", niz);
    niz[2] = 'e';
    niz[3] = 'k';
}

void f2(void) {
    niz[0] = 'Q';
    static char niz[MAXNIZ + 1] = "Ada";
    printf("%s!\n", niz);
    niz[0] = 'I';
}

void f3(void) {
    printf("%lf %lf\n", pTocka->x, pTocka->y);
    (++pTocka)->x++;
}

int main(void) {
    f1();
    printf("%s!\n", niz);
    f1();

    f2();
    printf("%s!\n", niz);
    f2();

    f3();
    f3();
    f3();

    return 0;
}
```

7. Načiniti "generator" Fibonaccijevih brojeva (članova Fibonaccijevog niza). U datoteci `fibonacci.c` napisati definicije funkcija čiji su prototipovi prikazani u nastavku:

```
void resetFibonacci(void);  
int getNextFibonacci(void);
```

Funkcija `getNextFibonacci` kod svakog poziva treba vratiti sljedeći član niza Fibonaccijevih brojeva:

1. poziv `getNextFibonacci()` → 1
2. poziv `getNextFibonacci()` → 1
3. poziv `getNextFibonacci()` → 2
4. poziv `getNextFibonacci()` → 3
5. poziv `getNextFibonacci()` → 5
- itd.

Funkcija `resetFibonacci` vraća (*resetira*) "generator" Fibonaccijevih brojeva u početno stanje. Nakon poziva funkcije `resetFibonacci`, uzastopnim pozivima funkcije `getNextFibonacci` ponovo će se početi dobivati članovi niza od početka, 1, 1, 2, 3, 5, ...

U datoteci `fibonacciMain.c` napisati glavni program koji će s tipkovnice učitati cijeli broj `n`. Ako je učitani broj veći od nule, pozivom funkcije `resetFibonacci` postaviti generator Fibonaccijevih brojeva u početno stanje te pomoću `n` uzastopnih poziva funkcije `getNextFibonacci` generirati i ispisati `n` prvih članova Fibonaccijevog niza. Ponavljati postupak (s tipkovnice učitati `n`, *resetirati* generator i ispisati `n` članova Fibonaccijevog niza) dok god se za `n` ne upiše broj manji ili jednak nuli.

Primjer izvršavanja programa.

Upisite broj Fibonaccijevih brojeva > 6↵	<code>resetFibonacci</code>
1↵	<code>getNextFibonacci → 1</code>
1↵	<code>getNextFibonacci → 1</code>
2↵	<code>getNextFibonacci → 2</code>
3↵	<code>getNextFibonacci → 3</code>
5↵	<code>getNextFibonacci → 5</code>
8↵	<code>getNextFibonacci → 8</code>
Upisite broj Fibonaccijevih brojeva > 3↵	<code>resetFibonacci</code>
1↵	<code>getNextFibonacci → 1</code>
1↵	<code>getNextFibonacci → 1</code>
2↵	<code>getNextFibonacci → 2</code>
Upisite broj Fibonaccijevih brojeva > 0↵	<code>prekini daljnje učitavanje n</code>

8. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
void fun1(void);
void fun2(void);
void fun3(void);
void fun4(void);
```

proto.h

```
#include <stdio.h>
#include "proto.h"
extern int x;

int main(void) {
    int x = 30;
    x += 2;
    printf("%d\n", x);
    fun1();
    fun2();
    fun3();
    fun4();
    fun3();
    return 0;
}

void fun1(void) {
    x += 3;
    printf("%d\n", x);
}
```

glavni.c

```
#include <stdio.h>
#include "proto.h"
int x = 20;

void fun2(void) {
    x += 4;
    printf("%d\n", x);
}
```

modulA.c

```
#include <stdio.h>
#include "proto.h"

void fun3(void) {
    static int x = 5;
    x += 5;
    printf("%d\n", x);
}

void fun4(void) {
    extern int x;
    x += 6;
    printf("%d\n", x);
}
```

modulB.c

Prevođenje programa testirati na dva načina:

- jednim pozivom prevodioca prevesti i povezati sva tri modula
- svaki od modula prevesti zasebno po jednim pozivom prevodioca, a zatim povezati dobiveni objektni kod

Rješenja:

1. -

2. -

3. -

4. -

5. -

6. -

7. Datoteka **fibonacci.h**

```
void resetFibonacci(void);  
int getNextFibonacci(void);
```

Datoteka **fibonacci.c**

```
#include "fibonacci.h"  
  
static int f_minus_1 = 1, f_minus_2 = 1, rbr = 0;  
  
void resetFibonacci(void) {  
    f_minus_1 = 1;  
    f_minus_2 = 1;  
    rbr = 0;  
    return;  
}  
  
int getNextFibonacci(void) {  
    int nextFib;  
    ++rbr;  
    if (rbr <= 2) {  
        nextFib = 1;  
    } else {  
        nextFib = f_minus_1 + f_minus_2;  
        f_minus_2 = f_minus_1;  
        f_minus_1 = nextFib;  
    }  
    return nextFib;  
}
```

Datoteka **fibonacciMain.c**

```
#include <stdio.h>

#include "fibonacci.h"

int main(void) {
    int n;
    do {
        printf("Upisite broj Fibonaccijevih brojeva > ");
        scanf("%d", &n);
        if (n > 0) {
            resetFibonacci();
            for (int i = 0; i < n; ++i) {
                printf("%d\n", getNextFibonacci());
            }
        }
    } while (n > 0);
    return 0;
}
```

8. -