ZADACI

Definicija funkcije

1. Koje su ispravne definicije funkcije `definicija` ako funkcija ne vraća vrijednost i ne prima parametar:

a)
```
void definicija(){
	return;
}
```

b)
```
void definicija(){
}
```

c)
```
void definicija{
	return;
}
```

d)
```
void definicija(void){
	return;
}
```

e)
```
void definicija(void){
}
```

f)
```
void definicija{
}
```

g)
```
void definicija[]{
	return;
}
```

h)
```
void definicija[void]{
	return;
}
```

i)
```
definicija(){
	return;
}
```

Naredba return

Kojeg je tipa povratna vrijednost u funkciji `pv`?
```
int pv(float a){
	float b;
	b=2*a;
	return b;
}
```

Void funkcije i funkcije bez argumenata

Što će ispisati sljedeći program?

```
#include <stdio.h>

void f(int a, int b) {
  a++;
  b++;
  printf("%d %d", a, b);
}

int main() {
  int a=3, b=4;
  f(3,4);
  printf("\n%d %d", a, b);
  return 0;
}
```

Prijenos kopija vrijednosti (bez polja)

Što će se ispisati obavljanjem sljedećeg programa?

```
int funkcija (int x, int y) {
  x = x + y;
  return x * y;
}

int main () {
  int x = 2, y = 3;
  printf ("%d, %d, %d", x, y, funkcija (x, y));
  return 0;
}
```

Prijenos referencija-adresa (bez polja)

Što će se ispisati sljedećim programom ?

```
void f(int x, int *y) {
  x %= 2;
  *y *= x;
}

int main () {
  int a=8, b=10;
  printf ("%d %d", a, b);
  f (a, &b);
  printf (" -> %d %d", a, b);
  return 0;
}
```

Jednodimenzionalna polja kao argumenti funkcije (rad s pokazivačima)

Pretpostavimo da funkcija `suma` računa sumu članova polja duljine N.
Koji od navedenih je pravilan poziv funkcije iz glavnog programa ako želimo izračunati sumu članova polja podaci?

```
float suma(float *polje, int n){
     (…)
}
int main(){
     float podaci[]={1, 1.4, 2, 3.5};
     float s;

     //poziv funkcije..

     return 0;
}
```

```
1) s=suma(*podaci[0],4);
2) s=suma(*podaci,4);
3) s=suma(&podaci,4);
4) s=suma(&podaci[0],4);
5) s=suma(podaci[0],4);
```

Smještajni razredi (postojanost, područje važenja varijabli) Samo elementarni pojmovi, po mogućnosti bez register i external!

Ukoliko vrijednost lokalne varijable treba biti zapamćena po izlasku i ponovnom povratku u funkciju kojem smještajnom razredu treba pripadati ta varijabla?

Dvodimenzionalna polja kao argumenti funkcije (rad s indeksnim izrazima)

Deklarirana je funkcija koja prima dvodimenzionalno polje kao argument:

```
int func( int p[], int m, int n, int maxstup )
```

Ukoliko je varijabli a potrebno pridružiti element polja s indeksom retka 7 i indeksom stupca 5, upotrijebit ćemo sljedeću naredbu.

```
1) a = p[5 * maxstup + 7];
2) a = p[7][5];
3) a = p[7 * maxstup + 5];
4) a = p[5][7];
5) a = p[7*5];
```

Dvodimenzionalna polje kao argument funkcije (rad s pokazivačima)

Ukoliko funkcija f treba izračunati sumu svih elemenata u matrici koju naredbu treba umetnuti na mjesto označeno s ### ?

```
int f(int *p, int m, int n, int maxstup){
        int i,j,suma=0;
        for(i=0; i<m ; i++){
              for(j=0; j<n; j++){
                    suma += *p;
                    p++;
              }
              ###
        }
        return suma;
}
```

Jednodimenzionalna polja kao argumenti funkcije (rad s indeksnim izrazima)

Napraviti funkciju koja provjerava jesu li elementi nekog cjelobrojnog polja fibonnacijevi brojevi. Funkcija mora imati prototip:

```
int provjeraFib(int [], int);
```

Formalni i stvarni argumenti (izrazi, redoslijed, tipovi pri pozivu funkcije)

Prototip funkcije je: `void f2 (int x, float y, char z);`
Koji su od ponuđenih poziva funkcije **ispravni**?

1. `f2('A', "Abc", 'B');`
2. `void f2('A', 15.01, 'B');`
3. `f2('A', 15.01, 'B');`
4. `f2(23, 0.1f, 65);`
5. `int i; i = f('A', 56.f, 'z');`

Prototipovi funkcija, organizacija složenijih programa

Koji su od navedenih prototipova funkcije `traziMax` **ispravni**?

```
int traziMax(int p[], int duljina) {
   int i = 0, max = p[0];
   for (i = 1; i < duljina; i++) {
      if (p[i] > max) {
         max = p[i];
      }
   }
   return max;
}
```

1. `int traziMax(int *p, int duljina);`
2. `int traziMax(int p, int duljina);`
3. `void traziMax(int *p, int duljina);`
4. `int traziMax(int *p, int duljina, int max);`
5. `int traziMax(int p[], int duljina);`

RJEŠENJA
Definicija funkcije
a, b, d, e

Naredba return
Povratna vrijednost je tipa `int.`

Void funkcije i funkcije bez argumenata
4 5
3 4

Prijenos kopija vrijednosti (bez polja)
`2, 3, 15`

Prijenos referencija-adresa (bez polja)
8 10 -> 8 0

Jednodimenzionalna polja kao argumenti funkcije (rad s pokazivačima)
4

Smještajni razredi (postojanost, područje važenja varijabli) Samo elementarni pojmovi, po mogućnosti bez register i external!

Odgovor: `static`

Dvodimenzionalna polja kao argumenti funkcije (rad s indeksnim izrazima)
3

Dvodimenzionalna polje kao argument funkcije (rad s pokazivačima)
p += maxstup – n;

Formalni i stvarni argumenti (izrazi, redoslijed, tipovi pri pozivu funkcije)
Rješenje: 3, 4

Prototipovi funkcija, organizacija složenijih programa
Rješenje: 1, 5

Jednodimenzionalna polja kao argumenti funkcije (rad s indeksnim izrazima)

```
#include <stdio.h>
#define N 10

int provjeraFib(int [], int);
void ispisPolja(int [], int);

int main()
{
      int poljeFib[N] = {1, 1, 2, 3, 5, 8, 13, 21, 34, 55};
      int poljeNeFib[N] = {1, 1, 2, 3, 5, 8, 13, 22, 34, 55};

      printf("Polje: ");
```

```
	ispisPolja(poljeFib, N);
	if(provjeraFib(poljeFib, N)) printf(" JE polje fibinaccijevih
brojeva\n");
	else printf(" NIJE polje fibinaccijevih brojeva\n");

	printf("Polje: ");
	ispisPolja(poljeNeFib, N);
	if(provjeraFib(poljeNeFib, N)) printf(" JE polje fibinaccijevih
brojeva\n");
	else printf(" NIJE polje fibinaccijevih brojeva\n");

	return 0;
}

void ispisPolja(int polje[], int n)
{
	int i;
	for(i=0; i<n; i++)
	{
		printf("%d ", polje[i]);
	}
}

int provjeraFib(int polje [], int n)
{
	int i;
	if(polje[0] != 1 || polje[1] != 1) return 0;
	for(i=2; i<n; i++)
	{
		if(polje[i] != polje[i-1] + polje[i-2])
		{
			return 0;
		}
	}
	return 1;
}
```