DUM 20 téma: Test č. 3

ze sady: 2 **tematický okruh sady:** Vyšší programovací jazyky
ze šablony: 10 – Algoritmizace a programování **určeno pro:** 1. a 2. ročník
vzdělávací obor: 18-20-M/01 Informační technologie
26-41-M/01 Elektrotechnika - Elektronické počítačové systémy
vzdělávací oblast: odborné vzdělávání
metodický list/anotace: VY_32_INOVACE_10220ml.pdf

Nejprve vyřešte všechny úlohy, a následně odpovědi zaznamenejte do záznamového archu. Správnou odpověď označte křížkem.

Záznamový arch:

I. Pointery

	A	B	C	D	E	F	G
1							
2							
3							
4							

II. Pointery

	A	B	C	D	E	F	G
1							
2							
3							

III. Struktury

	A	B	C	D	E	F	G
1							
2							
3							
4							

IV. Dynamické proměnné

	A	B	C	D	E	F	G
1							
2							
3							
4							

V. OOP

	A	B	C	D	E	F	G	H
1								
2								
3								
4								

VI. OOP

	T	F
1		
2		
3		
4		

I. Pointery

Přiřaďte správné hodnoty na neznámá místa v tabulce za předpokladu, že adresa proměnné i=9 a adresa proměnné p=10:

```
int i,*p;
```

	i	p
p=&i;		
i=7;	1.	
*p=5;	2.	
*p=i+1;	3.	4.

A. 5
B. 6
C. 7
D. 8
E. 9
F. 10
G. jiná možnost

II. Pointery jako parametry funkce

```
1. int fce1(char *x){
     int i=0;
     while(x[i]!='\0')
       i++;
     return i;
   }
2. void fce2(char *x){
     int i=0;
     while(x[i]!='\0')
       if(x[i]=' ')
         x[i]='\0';
       else
         i++;
   }
```

```
3. void fce3(char *x){
    int i=0;
    while(x[i]!='\0'){
      if(x[i]>='a'&&x[i]<='z')
          x[i]=x[i]-'a'+'A';
          i++;
      }
   }
```

Ke každé funkci (fce1, fce2, fce3) přiřaďte, co dělá:

A. Zjistí počet čísel v řetězci.
B. Zjistí počet mezer v řetězci.
C. Zjistí počet znaků v řetězci.
D. Odstraní z řetězce mezery.
E. Odstraní z řetězce malá písmenka.
F. Ořízne řetězec za první mezerou.
G. Žádná z předchozích možností.

III. Strukturované datové typy

```
typedef struct{
  int cit,jm;
}ZLOMEK;
```

```
___1.___ soucet(___2.___){
  ___3.___
  c.cit=a.cit*b.cit;
  c.jm=a.jm*b.jm;
  return ___4.___
}
```

Přiřaďte, aby funkce `soucet` **sčítala dva zlomky:**

A. int c;
B. ZLOMEK c;
C. ZLOMEK
D. a;
E. c;
F. ZLOMEK a, ZLOMEK b
G. jiná možnost

IV. Dynamicky alokované proměnné

```
1. typedef struct PRVEK{
       int hod;
       PRVEK *dalsi;
   }PRVEK;
```

```
___1.___ odeber(PRVEK *x){
  PRVEK *pom;
  ___2.___;
  ___3.___;
  ___4.___;
}
```

Přiřaďte, aby funkce `odeber` odebrala první prvek v lineárním spojovém seznamu:

```
A. delete x;
B. delete pom;
C. pom=x->dalsi;
D. x=pom->dalsi;
E. return pom;
F. PRVEK *
G. void
```

V. Objektově orientované programování

Přiřaďte k jednotlivým pojmům, na kterém řádku se nacházejí/začínají:

1. Definice objektu
2. Definice třídy
3. Definice metody
4. Konstruktor

```
A. class kruznice{
B.  public:
C.   double s_x,s_y,r;
D.   void vypis(){
E.      printf("[%lf,%lf],%lf",s_x,s_y,r);}
F.   kruznice(double a,double b,double c){
G.      s_x=a;s_y=b,r=c}
   }; ...
H. kruznice k(1,1,4);
```

VI. Objektově orientované programování

Rozhodněte, která tvrzení jsou pravdivá a která jsou nepravdivá:

1. Předek má všechny vlastnosti (data a metody) potomka a nějaké další vlastnosti navíc.
2. Polymorfismus umožňuje, aby se stejné metody u různých tříd prováděly jinak.
3. Konstruktor je speciální metoda, kterou musí mít každá třída, aby bylo možné vytvořit objekt dané třídy.
4. Data a metody v části `private` může použít pouze programátor, který je naprogramoval a jsou tak chráněna proti hackerům.