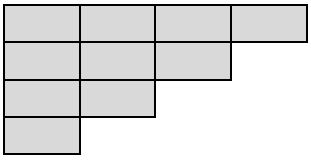
Uwaga – wszystkie programy (funkcje) nie powinny zawierać zbędnych instrukcji!

Zadanie 1

Korzystając z funkcji **`malloc()`** napisz funkcję **`alokuj()`**, która alokuje pamięć dla tablicy trójkątnej o N wierszach. Przyjmując, że numerujemy wiersze od zera, k – ty wiersz powinien składać się z $N - k$ kolumn (dla $N \geq 1$ i $k \in [0, N-1]$). Poglądowo, dla $N = 4$ alokowana pamięć, dla wspominanej tablicy powinna wyglądać następująco:



Funkcja **`alokuj()`** powinna pobierać jeden argument w postaci liczby całkowitej oznaczającej liczbę wierszy wspomnianej tablicy i zwracać wskaźnik do zaalokowanego bloku pamięci.

Zadanie 2

Napisz funkcję **`wypelnij_wyswietl()`**, która wypełni każdy wiersz opisanej w *zadaniu 1* tablicy kolejnymi liczbami naturalnymi poczynając od 1, a następnie wyświetli jej zawartość wiersz po wierszu. Poglądowo, dla $N = 4$ wypełniona tablica powinna wyglądać następująco:

1	2	3	4
1	2	3	
1	2		
1			

Zadanie 3

Napisz funkcję **`liczba_podciagow()`**, która pobiera dwa wskaźniki łańcuchowe. Funkcja powinna zliczać i zwracać liczbę wystąpień drugiego łańcucha w łańcuchu pierwszym. Uwaga: funkcja powinna zliczać tylko wystąpienia łańcuchów składających się z różnych znaków (różnych co do pozycji) tzn. wywołanie funkcji **`liczba_podciagow("mamama","mam")`** powinno zwracać 1 a nie 2.

Zadanie 4

Na stronie nr 2 znajduje się fragment programu (***baza.c***), w którym brakuje jednak trzech definicji i prototypów funkcji. Napisz odpowiedni prototyp funkcji **`inicjalizacja_OSOBA()`**, która dla podanych argumentów (zob. wywołanie funkcji w programie) inicjalizuje strukturę odpowiednimi danymi. Wykorzystaj funkcję **`malloc`**(), aby przydzielić odpowiednią ilość miejsca w strukturze na przechowanie łańcuchów oznaczających imię, nazwisko i miejscowość (przydzielone bloki pamięci powinny być dokładnie takiej wielkości jak podane dane przez użytkownika + jedno miejsce na znak pusty '\0').

Zadanie 5

Napisz odpowiednie prototypy i definicje kolejnych dwóch funkcji użytych w programie ***baza.c*** (**`wyświetl()`** i **`dealokacja_OSOBA()`**). Pierwsza z tych funkcji powinna wyświetlać strukturę, której adres jest jej przekazany jako argument. Druga z tych funkcji powinna dealokować pamięć przydzieloną (przez funkcję **`inicjalizacja_OSOBA()`**) dla pierwszych trzech pól struktury, której adres pobiera jako swój argument. Napisz alternatywną definicję funkcji **`wyświetl()`**, do której struktura jest przekazywana poprzez wartość a nie wskaźnik. Czy da się w podobny sposób napisać alternatywną definicję dla funkcji **`dealokacja_OSOBA()`**?

PROGRAM *baza.c*

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define N 30
#define MAX 10
typedef struct osoba{
	char *imie;
	char *nazwisko;
	char *miejscowosc;
	int wiek;
} OSOBA;
.....
.....
.....
int main(void)
{
	OSOBA dane[MAX];
	char tab_imie[N+1];
	char tab_nazwisko[N+1];
	char tab_miejscowosc[N+1];
	int i = 0, age, k;
	while (i < MAX)
	{
		printf("Podaj imie: ");
		gets(tab_imie);
		printf("Podaj nazwisko: ");
		gets(tab_nazwisko);
		printf("Podaj miejscowosc: ");
		gets(tab_miejscowosc);
		printf("Podaj wiek: ");
		scanf("%d", &age);
		while(getchar() != '\n');
		inicjalizacja_OSOBA(&dane[i], tab_imie, tab_nazwisko, tab_miejscowosc, age);
		i++;
	}
	printf("Podaj, ktory rekord tablicy dane chcesz wyswietlic.\n");
	printf("Wyjscie poza indeks tablicy lub wpisanie znaku nie bedacego liczba konczy
wypisywanie!");
	while (scanf("%d", &k) == 1 && k >= 0 && k < MAX)
	{
		printf("Rekord %d\n", k);
		printf("--------------------------\n");
		wyswietl(&dane[k]);
		printf("--------------------------\n");
		while(getchar() != '\n');
		printf("Podaj ktory rekord tablicy dane chcesz wyswietlic.\n");
		printf("Wyjscie poza idkeks tablicy lub wpisanie znku nie bedacego liczba konczy
wypisywanie!\n");
	}
	for(int i = 0; i < MAX; i++)
		dealokacja_OSOBA(&dane[k]);
	printf("KONIEC!");

	return 0;
}
```

Zadanie 6

Poniżej znajduje się niekompletny program uzupełnij go o fragmenty kodu, w których znajdą się wywołania odpowiednich funkcji, tak aby program wykonywał następujące zadanie:

1. program powinien otworzyć plik (sprawdzając czy nie wystąpił błąd otwarcia pliku) do zapisu i odczytu , którego nazwa będzie podana jako argument wiersza poleceń;
2. program powinien zapisać strukturę **`produkt`** do otwartego pliku;
3. program powinien odczytać strukturę z pliku i zapisać ją pod zmienną **`kopia_produkt`**
4. program powinien wyświetlić strukturę **`kopia_produkt`** i zamknąć plik (sprawdzając czy nie wystąpił błąd zamknięcia pliku).

```
#include<stdio.h>
#include<stdlib.h>
#define N 21
typedef struct towar{
      unsigned int id;
      char nazwa[N];
      float cena;
} TOWAR;
int main(.....)
{
      TOWAR produkt = {.nazwa = "pralka", .cena = 899.99, .id = 1};
      TOWAR kopia_produkt;
      ..........................................................
      ..........................................................
      ..........................................................
      ..........................................................
      ..........................................................
      ..........................................................
      ..........................................................
      ..........................................................
      ..........................................................
      ..........................................................
      printf("id: %u  nazwa: %s  cena %.2f zl",
       kopia_produkt.id, kopia_produkt.nazwa, kopia_produkt.cena);
      ..........................................................
      ..........................................................
      ..........................................................

      return 0;
}
```

Zadanie 7

Zaprojektuj funkcję , która wypełnia i wyświetla tablicę dwuwymiarową o rozmiarze **W x K** - uzupełnij podany program o stosowny prototyp i definicję funkcji. Tablica powinna być wypełniona losowymi małymi i wielkimi literami alfabetu łacińskiego. Instrukcje wyświetlania należy tak zapisać, aby każdy wiersz tablicy oddzielony był znakiem nowej linii, a każda litera w wierszu spacją.

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define W 8
#define K 12
.....
int main (void){
      char tablica[W][K];
      uzupelnij_wyswietl(tablica);

      return 0;
}
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

Zadanie 8

Napisz funkcję **moja_atoi()**, która powinna być autorską wersją funkcji **atoi()** pobierającej łańcuch znaków i konwertującej go na odpowiednią liczbę całkowitą - wartość zwracana tej funkcji.

Przykładowe wywołania funkcji i jej wartości zwracane:

```
moja_atoi("123") -> 123
moja_atoi("-223") -> -223
moja_atoi("  23") -> 23
moja_atoi("- 223") -> 0
moja_atoi("a223") -> 0
moja_atoi("2ala") -> 2
moja_atoi("-5.4") -> -5
moja_atoi("   -5.4ola") -> -5
```

Zadanie 9

Napisz program, który otwiera do odczytu plik *lec.txt*, który zawiera 26 przykładowych wierszy tekstu różnej długości. Program powinien wywoływać funkcję **`alokuj_tab_str()`**, która pobiera wskaźnik plikowy. Funkcja **`alokuj_tab_str()`** powinna alokować pamięć dla tablicy 26 wierszy znaków (char) o długości o jeden większej od liczby znaków w kolejnych wierszach pliku. W tym celu funkcja **`alokuj_tab_str()`** powinna zliczać liczbę znaków w wierszu (bez znaku nowej linii) i alokować odpowiedni blok pamięci. Następnie funkcja powinna odczytywać kolejne wiersze pliku znak po znaku i zapisywać je w odpowiednich wierszach tablicy, dodając po ostatnim znaku odczytanego wiersza znak pusty (‘\0’). Funkcja **`alokuj_tab_str()`** powinna zwracać wskaźnik do tak stworzonej tablicy łańcuchów. W dalszej kolejności program powinien wypisywać kolejne wiersze tablicy używając funkcji **`puts()`** i dealokować przydzieloną pamięć.

Zadanie 10

Podać wartości, które zostaną wyświetlone na ekranie po wykonaniu poniższego programu. Zachować kolejność.

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

struct wezel{
   int dane;
   struct wezel  *nastepny;
};
int main(void)
{
 int k;
 struct wezel  *p,  *q;
 p=NULL;
 for( k=1; k<5; ++k){
   q= malloc(sizeof(struct wezel));
   q->dane=2*k;
   q->nastepny=p;
   p=q;
 }
 while(q!=NULL){
   printf("%d ", q->dane);
   q=q->nastepny;
 }
 return 0;
}
```

Zadanie 11

Podać wartości, które zostaną wyświetlone na ekranie po wykonaniu poniższego programu. Zachować kolejność.

```
#include <stdio.h>
void f(int x){
 if(x<3)f(x+1);
 printf("%d\n", x);
 if(x<5)f(x+2);
}

int main(void){
 f(2);
 return 0;
}
```

Zadanie 12

Podać wartości, które zostaną wyświetlone na ekranie po wykonaniu poniższego programu. Zachować kolejność.

```
#include <stdio.h>
void f(long  *p1, long  *p2, long x){
   while(p1!=p2){
      if( *p1>x)printf("%ld\n",  *p1);
      ++p1;
   }
}
int main(void){
 long a[15]={5, 9, 6, 1, 7, 4, 8, 2, 1, 3, 7, 6, 5, 1, 9};
 f(a+2, a+11, 5);
 return 0;
}
```

Zadanie 13

Jaką zawartość będzie miała tablica t po wykonaniu programu.

```
#include <stdio.h>
#include <string.h>
void f( char *s){
 while(*s){
  if(!(strlen(s)%2)) *s='!';
  ++s;
  }
}
int main(void){
 char t[20]="programowanie";
 f(t);
puts(t);
 return 0;
}
```

Zadanie 14

Podać wartości, które zostaną wyświetlone na ekranie po wykonaniu poniższego programu. Zachować kolejność i format wydruku.

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

void f(float *x, float *y) {
 *y=--*x; x=y; ++*x;
}

int main(void) {
 float *tab,*x;
 tab =(float *) malloc (2*sizeof(float));
 tab[0]=5;
 x=&*(tab+1);
 f(tab,x);
 printf ("%.1f %.2f",tab[0],tab[1]);
 return 0;
}
```

Zadanie 15

Jakie wartości zostaną wyświetlone po wykonaniu poniższego fragmentu programu:

```
int t[4]={2,5,-4,-3}, *p;
p=t+2; p++;
printf("%d\t %d\t %d\t %d\t %d\t", 2**p,  2**t+4,  2**(t+1),  4*(*p+2),
p-t);
```

Zadanie 16

Jakie wartości i w jakiej kolejności zostaną wyświetlone na ekranie po wykonaniu poniższego programu. Zachować podział na wiersze.

```
#include <stdio.h>
int f(int *y, int x){
int i, *z=y+2;
for(i=1; *(y+i)>x; i+=2)
  if(*z++>2)
   printf("%d %d \n", i, y[i]);
}
int main(void){
 int tab[8]={5,4,6,10,9,11,2,1};
 f(tab,2);
 return 0;
}
```

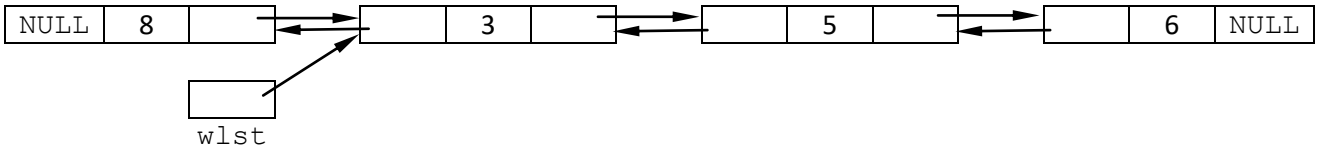
Zadanie 17

Co się wyświetli wyniku wykonania poniższego programu?

```
#include <stdio.h>
#include <string.h>
int main(){
   char *s1="programowanie ", s2[30]="programowanie proceduralne",
   *s3="program", *w;
   w=s2+strlen(s1);
   puts(w);
   if(!strcmp(s1,s3))
      strcpy(s2,s3);
   puts(s2);
   return 0;
}
```

Zadanie 18

Napisać deklarację struktury, która może być wykorzystana do utworzenia listy dwukierunkowej. Struktura zawiera składową całkowitą oraz dwa wskaźniki. Jeden ze wskaźników wskazuje następną strukturę listy, drugi poprzednią. Pierwsza i ostatnia struktura listy zawierają wskaźniki puste (znajdujący się w pierwszej strukturze listy wskaźnik przeznaczony do wskazania poprzedniej struktury listy oraz znajdujący się w ostatniej strukturze lisy wskaźnik, który ma wskazywać następną strukturę przyjmują wartość NULL). Jedna ze struktur listy jest wskazywana przez wskaźnik o nazwie wlst. Napisać deklarację tego wskaźnika.

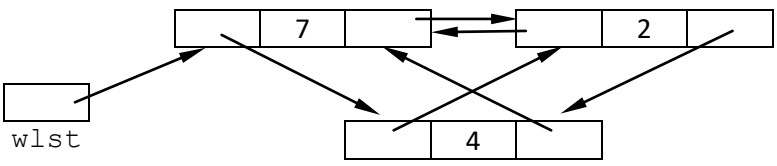


Napisać funkcję dodającą jedną strukturę do listy. Funkcja ma posiadać dwa parametry. Pierwszym argumentem funkcji wskaźnik wskazujący jedną ze struktur listy. Nowa struktura ma zostać umieszcza bezpośrednio przed strukturą wskazywaną przez ten wskaźnik. Drugi argument przekazuje wartość całkowitą, która będzie umieszczona w dodawanej do listy strukturze. Funkcja nie może zawierać zbędnych instrukcji. Nie może też niczego wyświetlać na ekranie.

Rysunek powyżej przedstawia przykładową listę. Funkcja powinna działać poprawnie dla listy zawierającej dowolną liczbę struktur przechowujących dowolne wartości całkowite.

Zadanie 19

Napisać instrukcje tworzące listę cykliczną przedstawioną na poniższym rysunku. Nie wolno zamieszczać zbędnych instrukcji. Pamięć dla wszystkich struktur ma być przydzielona dynamicznie. Można używać innych zmiennych (także wskaźników), ale należy zamieścić ich deklaracje.



Napisać instrukcje zwalniające pamięć zajmowaną przez struktury.

Zadanie 20

Plik program.exe został utworzony w wyniku kompilacji poniższego programu.

```
#include <stdio.h>
int main(int argc, char **argv)
{
 printf("%c %c %c\n", argv[2][0], argv[1][4], '9'-argc);
 printf("%s %c\n", argv[1], *argv[1]);
 return 0;
 }
```

Podać, co zostanie wyświetlone na ekranie po uruchomieniu tego programu poleceniem

```
program programowanie proceduralne
```

Przyjąć, że do kodowania znaków wykorzystywany jest kod ASCII.

Zadanie 21

Podać wartości, które zostaną wyświetlone na ekranie w wyniku wykonania poniższego programu. Zachować kolejność i format wydruku.

```
#include <stdio.h>
int x=5;
int f(int n){
 static int y=1;
 --y; x+=y;
 printf ("n=%d \t x=%d \t y=%d \n", n, x, y);
 if ( n < 4) return f(n+3)+300;
 return 0;
}
int main(void){
 printf("%d",f(0));
 return 0;
}
```

Zadanie 22

Podać wartości, które zostaną wyświetlone na ekranie w wyniku wykonania poniższego programu. Zachować kolejność i format wydruku.

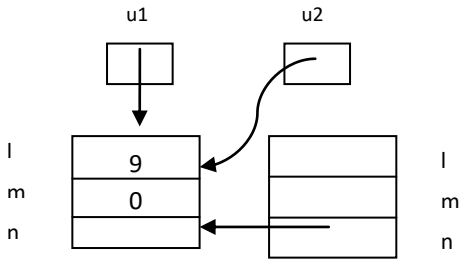
```
#include <stdio.h>
void f(int n){
 int x=0;
 static int y=1;
 --y; x+=y;
 if (n < 4) f(n+2);
 printf ("n=%d \t x=%d \t y=%d \n", n, x, y);
}
int main(void){
 f(0);
return 0;
}
```

Zadanie 23

W pewnym programie zdefiniowano następująco typy oraz zadeklarowano zmienne:

```
struct u{
  int l, m;
  struct u * n;
} *u1,*u2;
```

Dla każdej z poniższych sekwencji instrukcji sprawdź czy w wyniku jej wykonania otrzymamy następujący schemat. Zaznacz TAK lub NIE.



TAK	NIE	

		`u1=NULL; u2=u1; u1=(struct u *)malloc(sizeof(struct u));`
		`u2=(struct u *)malloc(sizeof(struct u));`
		`u2->n=u1; u2->n->l=9; u2->n->m=0; u1=u2;`

		`u1=NULL; u2=u1; u2=(struct u *)malloc(sizeof(struct u));`
		`u2->n=(struct u *)malloc(sizeof(struct u));`
		`u2=u2->n; u2->l=9; u2->m=0; u1=u2;`

		`u2=NULL; u1=u2; u1=(struct u *)malloc(sizeof(struct u));`
		`u1->n=(struct u *)malloc(sizeof(struct u));`
		`u2=u1; u2->l=9; u2->m=0;`

		`u2=NULL; u1=u2; u1=(struct u *)malloc(sizeof(struct u));`
		`u1->n=(struct u *)malloc(sizeof(struct u));`
		`u2=u1->n; u2->l=9; u2->m=0; u2=u1;`

Zadanie 24

Przy następujących deklaracjach:

```
int *p, **q;
int  (*a)[5];
int m1[2][5];
int m2[2][3];
```

Które z poniższych instrukcji są poprawne?

a. `p=m1[0];`
b. `p=&m2[1][2];`
c. `p=m1;`
d. `a=m1;`
e. `a=m2;`
f. `*q=m2[0];`
g. `q=m2;`

Zadanie 25

Podać wartości, które zostaną wyświetlone na ekranie w wyniku wykonania poniższego programu. Zachować kolejność wydruku.

```
#include <stdio.h>
void g(int x);
void f(int x){
 if(x>2)  g(x-4);
 printf("%d\t", x);
}
void g(int x){
 printf("%d\t", x);
 f(x+2);
}
int main(void){
 f(6);
 return 0;
}
```

Zadanie 26

Podać wartości, które zostaną wyświetlone na ekranie w wyniku wykonania poniższego programu. Zachować kolejność wydruku.

```
#include <stdio.h>
#define W 3
#define K 4

int sum(int *start , int *stop);

int main (void){
  int tab[W][K]={{2,6,4,5},{0,0,2,6},{2,4,3,7}};
  int *p, *q;
  p = &tab[0][3];
  q = &tab[2][2];

  printf("%d\t", sum(tab[0], p));
  printf("%d\t", sum(p, tab[2]));
  printf("%d\t", sum(&tab[2][0], q));
  printf("%d\t", sum(tab[1]+1, tab[1]+3));
  printf("%d\t", sum(p+3, q+1));

  return 0;
}

int sum(int *start, int *stop){
  int temp=0, *s;
  int i=0;
  for(s=start; s<=stop; s++)
     temp +=*s;
  return temp;
}
```

Zadanie 27

Co się wyświetli wyniku wykonania poniższego programu? Zachowaj kolejność i format wydruku.

```
#include <stdio.h>
int main(void){
 char *w="f-g-e-t-s", *x=w;
 while(*++x);
 do
   if(*--x!='-'){
   putchar(*x);
   putchar('|');
   puts(x);
 }
 while(x-w);
return 0;
}
```

Zadanie 28

Plik tekstowy „tekst.txt” zawiera następujące dane (cztery wiersze i wiersz pusty jako piąty):

```
komputer
Lublin
biurko
widowisko
```

Funkcja f zdefiniowana została następująco:

```
void f(char *str){
  static int x = 0;
  x += strlen(str)-1;
  printf("%d\n", x);
}
```

Podaj wartości, które wyświetlą się na ekranie po wykonaniu następującego fragmentu programu. Zachowaj format wydruku. Zakładamy, że `plik` jest otwartym strumieniem plikowym w trybie do odczytu pliku „tekst.txt”. Przyjąć, że podczas wykonywania operacji na plikach nie wystąpią żadne błędy, a potrzebne pliki nagłówkowe są dołączone do programu.

```
char tab[20];
while(fgets(tab, 19, plik)!= NULL){
  if (tab[0] < 'p')
     f(tab);
  fputs(tab, stdout);
}
```

Zadanie 29

W pewnym programie zdefiniowano strukturę:

```
struct my_struct {
    int x;
    int *t1[4];
};
```

Zapisano następujące deklaracje i instrukcje:

```
int t[4][3]={{8,2,3}, {14,9,1}, {11,13,6}, {17,9,5}};
struct my_struct s;

int (*q)[3];

q = t+1;
s.x = (int) sizeof(t)/sizeof(t[0]);
for(int i = 3; i >=0; i--)
    s.t1[i] = t[3-i];
```

Określ, jakie wartości mają poniższe wyrażenia (załóż, że wyrażania są wykonywane sekwencyjnie, tzn. po obliczeniu pierwszego obliczane jest drugie, itd.)?

wyrażenie	wartość
`s.x`	
`*s.t1[2]`	
`s.t1[1][1]`	
`*s.t1[3]`	
`*(*(q+2)+2)`	
`(*(q[-1]+1))++`	
`*(s.t1[3]+1)`	

Zadanie 30

Co zostanie wyświetlone w wyniku wykonania poniższego programu. Zachować kolejność i podział na wiersze.

```
#include <stdio.h>
void write( char *[], int);
int main(void){
 char *countries[5]={"Italy", "Malta", "Spain", "Greece", "Portugal"};
 write(countries,5);
 return 0;
}

void write( char *t[], int n){
int i;
puts(*(t+3));

for(i=0;i<n;++i)
  printf("%c\t", **(t+i));

printf("\n");

for(;n>0;n--)
  puts(t[n-1]);
}
```

Zadanie 31

W pewnym programie w oparciu o poniższe definicje utworzono listę jednokierunkową.

```
struct vect {
  float x,y;
};
struct l{
  struct vect v;
  struct l *n;
};
```

Która z poniższych definicji funkcji `add` pozwoli na poprawne wstawienie (dodanie) elementu `one` na koniec listy.

A.

```
struct l * add(struct l *e, struct vect one){
if(e==NULL){
   e=(struct l*)malloc(sizeof(struct l));
   e->v=one;
   e->n=NULL;
}
else
   e->n=add(e->n, one);
return e;
}
```

B.

```
struct l * add(struct l *e, struct vect one){
if(e==NULL){
   e=(struct l*)malloc(sizeof(struct vect));
   (*e).v=one;
   (*e).n=NULL;
}
else
   add((*e).n, one);
return e;
}
```

C.

```
struct l * add(struct l *e, struct vect one){
if(e==NULL){
   e=(struct l*)malloc(sizeof(struct l));
   e->v=one;
   e->n=NULL;
}
else
   e=add(e->n, one);
return e;
}
```

D.

```
struct l * add(struct l *e, struct vect one){
if(e!=NULL){
   e=(struct l*)malloc(sizeof(struct vect));
   e->v=one;
   e->n=NULL;
}
else
   e->n=add(e->n, one);
return e;
}
```

E.

```
struct l * add(struct l *e, struct vect one){
if(e!=NULL){
   e=(struct l*)malloc(sizeof(struct l));
   e->v=one;
   e->n=NULL;
}
else
   add(e->n, one);
return e;
}
```

F.

```
struct l * add(struct l *e, struct vect one){
if(e!=NULL){
   e=(struct l*)malloc(sizeof(struct l));
   (*e).v=one;
   (*e).n=NULL;
}
else
   (*e).n=add((*e).n, one);
return e;
}
```