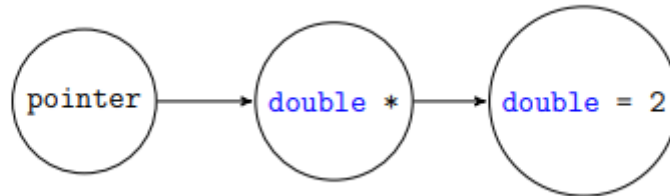


## Лабораторная работа №6

### 1) Тема: Указатели, арифметика указателей. Введение в функции.

6.1)

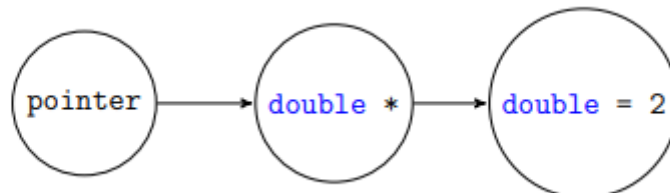
2) Внутри функции `main(void)` определите указатель `double **pointer = NULL;`. В оперативной памяти создайте конструкцию, показанную на рисунке 1.



При этом выполните следующее:

- используйте функции типа `*alloc(...)` для выделения оперативной памяти под динамические объекты;
- выведите число, указанное в крайней правой окружности, на экран, используя указатель `double **pointer = NULL;`
- используйте функцию `free(...)` для освобождения оперативной памяти, выделенную под динамические объекты.

3)



При этом выполните следующее:

- используйте функции типа `*alloc(...)` для выделения оперативной памяти под динамические объекты;
- выведите число, указанное в крайней правой окружности, на экран, используя указатель `double **pointer = NULL;`
- используйте функцию `free(...)` для освобождения оперативной памяти, выделенную под динамические объекты.

4)

Имя	Смысл	Тип
<code>*D</code>	Указатель на число	<code>double *</code>
<code>**DOUBLED</code>	Указатель на указатель	<code>double **</code>

5)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  int main() {
5      double *D;
6      double **DOUBLED;
7      D=(double *) malloc(sizeof(double));
8      if (D==NULL){
9          printf("ERROR 404");
10         return 1;
11     }
12     *D=2;
13     DOUBLED = (double **) malloc(sizeof(double *));
14     if (DOUBLED == NULL){
15         return 1;
16     }
17     DOUBLED=&D;
18     printf("%f\n", **DOUBLED);
19     free(DOUBLED);
20     return 0;
21 }
22
```

6)

```
2.000000
Process returned 0 (0x0)   execution time : 0.018 s
Press any key to continue.
```

6.2)

2) Напишите программу, которая складывает два числа с использованием указателей на эти числа.

3) D=&A;

Dade=&B;

C=\*D+\*Dade;

4)

Имя	Смысл	Тип
*D	Указатель на 1 число	Int *
*Dade	Указатель на 2 число	Int *
A	Число 1	Int
B	Число 2	Int
C	Сумма чисел	Int

5)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  int main() {
5      int *D;
6      int *DAde;
7      int A,B,C;
8      printf("LESSA GO!\n");
9      scanf("%d",&A);
10     scanf("%d",&B);
11     D=&A;
12     DAde=&B;
13     C=*D+*DAde;
14     printf("%d",C);
15     return 0;
16 }
17
```

6)

```
LESSA GO!
12
23
35
Process returned 0 (0x0)   execution time : 3.833 s
Press any key to continue.
```

6.3)

2) Напишите программу, которая находит максимальное число из двух чисел, используя указатели на эти числа.

3) if (\*D>\*DAde)

{printf("%d",\*D);}

If (\*D<\*DAde)

{printf("%d",\*DAde);}

4)

Имя	Смысл	Тип
*D	Указатель на 1 число	Int *
*DAde	Указатель на 2 число	Int *
A	Число 1	Int
B	Число 2	Int

5)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  int main() {
5      int *D;
6      int *DAde;
7      int A,B;
8      printf("LESSA GO!\n");
9      scanf("%d",&A);
10     scanf("%d",&B);
11     D=&A;
12     DAde=&B;
13     if (*D>*DAde)
14     {
15         printf("%d",*D);
16     }
17     if (*D<*DAde)
18     {
19         printf("%d",*DAde);
20     }
21     return 0;
22 }
23
```

6)

```
LESSA GO!
1
2
2
Process returned 0 (0x0)   execution time : 3.061 s
Press any key to continue.
```

6.4)

2) Напишите программу, которая создаёт одномерный динамический массив из чисел с плавающей точкой двойной точности, заполняет его значениями с клавиатуры и распечатывает все элементы этого массива, используя арифметику указателей (оператор +), а не обычный оператор доступа к элементу массива — [].

3)

p=A;

scanf("%d",&\*p);

p++;

scanf("%d",&\*p);

p++;

...

d=A;

printf(“%d”,\*d);

d++;

printf(“%d”,\*d);

d++;

...

4)

Имя	Смысл	Тип
A	Массив	Массив элементов типа int
*p	Указатель на элементы массива	Int *
*d	Указатель на элементы массива	Int *

5)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      int A[4];
6      int *p;
7      int *d;
8      p = A;
9      scanf ("%d", &*p);
10     p++;
11     scanf ("%d", &*p);
12     p++;
13     scanf ("%d", &*p);
14     p++;
15     scanf ("%d", &*p);
16     d=A;
17     printf ("%d", *d);
18     d++;
19     printf ("%d", *d);
20     d++;
21     printf ("%d", *d);
22     d++;
23     printf ("%d", *d);
24     return 0; }
25
```

6)

```
2
3
4
5
2345
Process returned 0 (0x0)   execution time : 3.930 s
Press any key to continue.
```

6.5)

2) Вычислить факториал заданного числа, используя указатель на целое число,  
а просто не переменную целого типа/

3) S=1;

p=A;

i=1;

while(i<=p)

{S=S\*i;

i+=1;}

4)

Имя	Смысл	Тип
A	Число, вводимое с клавиатуры	Int *
*p	Указатель на число	Int
i	Промежуточная переменная	Int
S	Переменная (значение факториала)	Int

5)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      int *p;
6      int i,A,S;
7      S=1;
8      scanf("%d",&A);
9      p=A;
10     i=1;
11     while(i<=p)
12     {
13         S=S*i;
14         i+=1;
15     }
16     printf("%d",S);
17     return 0;}
18
```

6)

```
4
24
Process returned 0 (0x0)   execution time : 2.764 s
Press any key to continue.
```

6.6)

2) Вывести элементы динамического массива целых чисел в обратном порядке, используя указатель и операцию декремента (--).

3)

```
i=2;
```

```
L=P;
```

```
scanf ("%d",&*L);
```

```
while (i<=N)
```

```
{
```

```
    L++;
```

```
    scanf ("%d",&*L);
```

```
    i++;
```

```
}
```

```
L=NULL;
```

```
L=P;
```

```
L=L+N-1;
```

```
i=N-2;
```

```
printf("%d",*L);
```

```
while (i>=0)
```

```
{
```

```
    L--;
```

```
    printf ("%d",*L);
```

```
    i--;
```

```
}
```

4)

Имя	Смысл	Тип
N	Размер массива	Int
i	Промежуточная переменная	Int
*P	Массив элементов	Int *
*L	Указатель на массив	Int *

5)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  main ()
5  {
6      int N,i;
7      int *P;
8      int *L;
9      scanf ("%d\n",&N);
10     P=(int *) malloc(N*sizeof(int));
11     i=2;
12     L=P;
13     scanf ("%d",&*L);
14     while (i<=N)
15     {
16         L++;
17         scanf ("%d",&*L);
18         i++;
19     }
20     L=NULL;
21     L=P;
22     L=L+N-1;
23     i=N-2;
24     printf ("%d",*L);
25     while (i>=0)
26     {
27         L--;\
28         printf ("%d",*L);
29         i--;
30     }
31     return 0;
32 }
33
```

6)

```
4
1
2
3
4
4321
Process returned 0 (0x0)   execution time : 7.442 s
Press any key to continue.
```



6.7) –

68)

2) Определите переменную целого типа `int a = 10;` и выведите побайтово её содержимое на экран, используя указатель `char *`;

3)

```
int a=10;
```

```
char *P;
```

```
P=(char *) &a;
```

```
S=sizeof(int);
```

```
While (S--)
```

```
{printf ("%d",*P);
```

```
P++;}
```

4)

Имя	Смысл	Тип
S	Промежуточная переменная	Int
a	Переменная целого типа	Int
*P	Указатель на переменную	Char *

5)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include <math.h>
4  int main()
5  {
6      int S;
7      int a=10;
8      char *P;
9      P=(char *) &a;
10     S = sizeof(int);
11     while (S--)
12     { printf ("%d",*P);
13       P++;
14     }
15     return 0;
16 }
17
```

6)

```
10000
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.
```

6.9)

2) Выделите память под двумерный динамический массив — матрицу — таким образом, чтобы данные все строки этой матрицы гарантированно располагались в оперативной памяти друг за другом

3)

```
P = (int **) malloc (R*sizeof(int*));
```

```
for (i=0; i<R; i++)
```

```
{ for (j=0; j<R; j++)
```

```
{ scanf("%d",&P[i][j]); }}
```

4)

Имя	Смысл	Тип
**P	Двумерный динамический массив	
P[i][j]	Элемент двумерного массива	
i	Индекс элемента	
j	Индекс элемента	
R	Кол-во строк	
C	Кол-во столбцов	

5)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include <math.h>
4  int main()
5  {
6      int i,R,C,j;
7      int **P;
8      scanf("%d",&R);
9      scanf ("%d",&C);
10     P = (int **) malloc(R*sizeof(int*));
11     for (i=0; i<R; i++)
12     {
13         P[i]=(int *) malloc(C*sizeof(int));
14     }
15     for (i=0;i<R;i++)
16     {
17         for (j=0;j<C;j++)
18         {
19             scanf("%d",&P[i][j]);
20         }
21     }
22     for (i=0;i<R;i++)
23     {
24         for (j=0;j<C;j++)
25         {
26             printf("P[%d][%d]=%d ",i,j,P[i][j]);
27         }
28     }
29     return 0;
30 }
31
```

6)

```
3
3
1
2
3
4
5
6
7
8
9
P[0][0]=1 P[0][1]=2 P[0][2]=3 P[1][0]=4 P[1][1]=5 P[1][2]=6 P[2][0]=7 P[2][1]=8 P[2][2]=9
Process returned 0 (0x0)   execution time : 9.462 s
Press any key to continue.
```

7.1)

2) Создайте две функции, которые вычисляют факториал числа, используя цикл:

- функцию, которая вычисляет факториал, используя цикл;
- функцию, которая вычисляет факториал, используя рекурсивный вызов самой себя

3)

a) int fact(int x)

{int p=1;

int g=1;

for (g=1;g<x+1;g++)

{p\*=g;}

return (p);

b) int fact (int n)

{if n==0

{return (1);}

int s;

s=fact(n-1)\*n;

return(s);

}

4)

Имя	Смысл	Тип
z	Значение факториала	Int
x	Число	Int
fact	1-ая Функция, рассчитывающая факториал числа	Int fact (int x)
x	Переменная функции	Int
p	Переменная функции	Int
g	Переменная функции	Int
fact	2-ая Функция, рассчитывающая факториал числа	Int fact (int n)
n	Переменная функции	Int
s	Переменная функции	Int
l	Число	Int
r	Значение факториала	Int

5\ a)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  int fact(int x)
5  {
6      int p=1;
7      int g=1;
8      for (g=1;g<x+1;g++)
9      {
10         p*=g;
11     }
12     return(p) ;
13 }
14 int main()
15 {int z,x;
16  scanf ("%d",&x) ;
17  z=fact(x) ;
18  printf ("%d",z) ;
19  return 0;
20 }
21
```

6\ a)

```
3
6
Process returned 0 (0x0)   execution time : 1.586 s
Press any key to continue.
```

5\ b)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  int fact(int n)
5  {
6      if (n==0)
7      {
8          return (1);
9      }
10     int s;
11     s=fact(n-1)*n;
12     return(s) ;
13 }
14 int main()
15 {int l,r;
16  scanf ("%d",&l) ;
17  r=fact(l) ;
18  printf ("%d",r) ;
19  return 0;
20 }
21
```

6\b)

```
4
24
Process returned 0 (0x0)   execution time : 1.179 s
Press any key to continue.
```

7.2)

2) Напишите отдельные функции для вычисления:

- количество размещений из n по k:  $A_n^k = n!/(n-k)!$  ;
- количество сочетаний из n по k:  $C_n^k = n!/k!(n-k)!$  .

3)

$A_n^k = n!/(n-k)!$  ;

$C_n^k = n!/k!(n-k)!$ ;

4)

Имя	Смысл	Тип
fact	Функция, рассчитывающая факториалы	Int fact (int n)
A	Функция, для расчёта размещений	Double A (int q, int o)
C	Функция, для расчёта сочетаний	Double C (int d, int k)
n	Переменная функции fact	Int
s	Значение функции fact	Int
w	Значение функции A	Double
q	Переменная функции A	Int
o	Переменная функции A	Int
d	Переменная функции C	Int
k	Переменная функции C	Int
p	Значение функции C	Double
z	Число n	Int
x	Число k	Int
s	Значение размещений из n по k	Float
f	Значение сочетаний из n по k	Float

5)

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

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

```
#include<math.h>
```

```
int fact(int n)
```

```
{
```

```
    if (n==0)
```

```

    {
        return (1);
    }
    int s;
    s=fact(n-1)*n;
    return(s);
}
double A(int q,int o)
{
    double w;
    w=fact(q)/fact(q-o);
    return (w);
}
double C (int d,int k)
{
    double p;
    p=fact(d)/(fact(k)*fact(d-k));
    return (p);
}
int main()
{int z,x;
float s,f;
scanf("%d",&z);
scanf("%d",&x);
s=A(z,x);
f=C(z,x);
printf("%f\n",s);
printf("%f",f);
return 0;
}

```

6)

```
3
2
6.000000
3.000000
Process returned 0 (0x0)   execution time : 2.811 s
Press any key to continue.
```

7.3)

2) Объявите указатель на массив типа `int` и динамически выделите память для 12-ти элементов. Напишите функцию, которая поменяет значения четных и нечетных ячеек массива.

3)

```
int change(int a){
    S=C;
    for(int i=0; i<(a/2); i++){
        int n = *S;
        S++;
        int k=*S;
        *S=n;
        S--;
        *S=k;
        S=S+2;
    }
}
```

4)

Имя	Смысл	Тип
*S	Указатель на массив	
*C	Массив	
change	Функция, меняющая местами чётные и нечётные ячейки массива	
i	Индексная переменная функции	
a	Переменная функции	
n	Переменная функции	
k	Переменная функции	
i	Индекс элемента массива	



5)

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

```
#include <math.h>
```

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

```
#include <locale.h>
```

```
#include <malloc.h>
```

```
int *S, *C;
```

```
int change(int a){
```

```
    S=C;
```

```
    for(int i=0; i<(a/2); i++){
```

```
        int n = *S;
```

```
        S++;
```

```
        int k=*S;
```

```
        *S=n;
```

```
        S--;
```

```
        *S=k;
```

```
        S=S+2;
```

```
    }
```

```
}
```

```
int main(){
```

```
    C = (int *) malloc(12 * sizeof(int *));
```

```
    S=C;
```

```
    for(int i=0; i<12; i++){
```

```
        scanf("%d",&*S);
```

```
        S++;
```

```
    }
```

```
    change(12);
```

```
    S=C;
```

```
    for(int i=0; i<12; i++){
```

```
printf("%d ", *S);
```

```
S++;
```

```
}
```

```
return 0;
```

```
}
```

6)

```
1
2
3
4
5
6
7
8
9
10
11
12
2 1 4 3 6 5 8 7 10 9 12 11
Process returned 0 (0x0)   execution time : 8.563 s
Press any key to continue.
```

7.4)

2) Создать две основные функции:

- функцию для динамического выделения памяти под двумерный динамический массив типа double — матрицу;
- функцию для динамического освобождения памяти под двумерный динамический массив типа double — матрицу.

Создать две вспомогательные функции:

- функцию для заполнения матрицы типа double;
- функцию для распечатки этой матрицы на экране.

3)

```
double** axe(int rows,int coll)
```

```
{
```

```
    double **A;
```

```
    A=(double **) malloc (rows * sizeof(double *));
```

```
    for (int i=0;i<rows;i++)
```

```
    {
```

```

        A[i]=(double *) malloc(coll*sizeof(double));
    }
    return A;
}
void FF (double **A,int rows)
{
    for (int i=0; i<rows;i++)
    {
        free(A[i]);
    }
    free(A);
}
void filup (double **A,int rows,int coll)
{
    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<coll;j++)
        {
            scanf("%d",&A[i][j]);
        }
    }
}
void pmd (double **A, unsigned rows, unsigned coll)
{
    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<coll;j++)
        {
            printf("%d\n",A[i][j]);
        }
    }
}

```

}

}

4)

Имя	Смысл	Тип
axe	Функция, выделяющая память под двумерный массив	double** axe(int rows,int coll)
FF	Функция, освобождающая память	void FF (double **A,int rows)
filup	Функция, заполняющая двумерный массив	void filup (double **A,int rows,int coll)
pmd	Функция, распечатывающая двумерный массив	void pmd (double **A, unsigned rows, unsigned coll)
**A	Матрица, используемая в функциях	double **
A[i][j]	Элемент матрицы, используемый в функциях	double
i	Индекс элемента матрицы, используемый в функциях	Int
j	Индекс элемента матрицы, используемый в функциях	Int
rows	Кол-во строк матрицы, используемое в функциях	Int
coll	Кол-во столбцов матрицы, используемое в функциях	Int
x	Количество строк матрицы	Int
z	Количество столбцов матрицы	Int
**p	Матрица	Double **
*locale	Локаль	Char *

5)

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

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

```
#include<math.h>
```

```
#include <locale.h>
```

```
double** axe(int rows,int coll)
```

```
{
```

```
    double **A;
```

```
    A=(double **) malloc (rows * sizeof(double *));
```

```

    for (int i=0;i<rows;i++)
    {
        A[i]=(double *) malloc(coll*sizeof(double));
    }
    return A;
}

void FF (double **A,int rows)
{
    for (int i=0; i<rows;i++)
    {
        free(A[i]);
    }
    free(A);
}

void filup (double **A,int rows,int coll)
{
    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<coll;j++)
        {
            scanf("%d",&A[i][j]);
        }
    }
}

void pmd (double **A, unsigned rows, unsigned coll)
{
    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<coll;j++)
        {

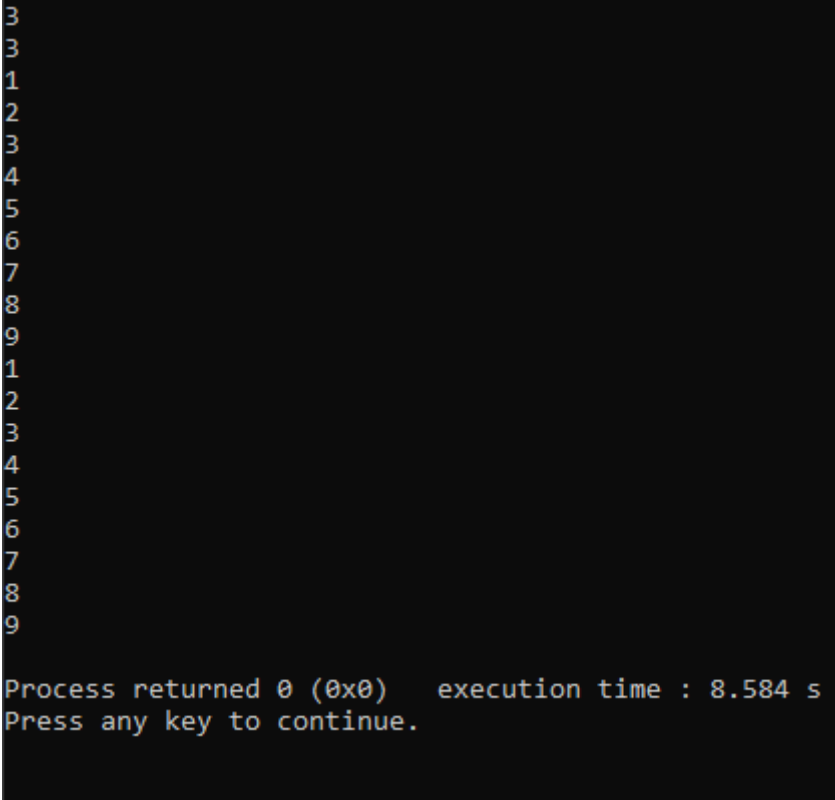
```

```

        printf("%d\n",A[i][j]);
    }
}
}
int main()
{
    char *locale=setlocale(LC_ALL, "");
    int x,z;
    scanf ("%d",&x);
    scanf ("%d",&z);
    double **p=axe(x,z);
    filup(p,x,z);
    pmd(p,x,z);
    FF(p,x);
    return 0;
}

```

6)



```

3
3
1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7
8
9

Process returned 0 (0x0)   execution time : 8.584 s
Press any key to continue.

```

7.5)

2) Создать две функции для динамических массивов (для указателей):

- функцию умножения матрицы типа double на вектор-столбец типа double;
- функцию умножения одной матрицы типа double на другую матрицу типа double.

3)

a) double\* v(double\*\*A, double\*B, int n,int k){

double \*C;

C=(double \*) malloc(n\*sizeof(double));

for (int i=0;i<n;i++)

{

for (int j=0;j<k;j++)

{

C[i]+=A[i][j]\*B[j];

}

}

return(C);

}

b) double\*\* v(double\*\*A, double \*\*B, int n,int k){

double \*\*C;

C=(double \*\*) malloc (n \* sizeof(double \*));

for (int i=0;i<n;i++)

{

C[i]=(double \*) malloc(k\*sizeof(double));

}

for(int i = 0; i < n; ++i) {

for(int j = 0; j < k; ++j) {

```

    C[i][j] = 0;
    for(int l = 0; l < n; ++l)
        C[i][j] += A[i][l] * B[l][j];
    }
}
return(C);
}

```

4)

a)

Имя	Смысл	Тип
v	Функция умножения матрицы на столбец	double* v(double**A, double*B, int n,int k)
**A	Матрица, используемая в функции	double**
*B	Столбец, используемый в функции	double*
n	Переменная функции	Int
k	Переменная функции	Int
*C	Столбец, выдаваемый функцией	double*
i	Индекс элементов массива	Int
j	Индекс элементов массива	Int
P	Количество строк	Int
L	Количество столбцов	Int
*locale	Локаль	char*
**S	Матрица	double**
*F	Столбец	double*
*R	Результат применения функции	double*



b)

Имя	Смысл	Тип
v	Функция умножения матриц	double* v(double**A, double*B, int n,int k)
**A	Матрица, используемая в функции	double**
**B	Матрица, используемая в функции	double**
n	Переменная функции	Int
k	Переменная функции	Int
**C	Матрица, выдаваемая функцией	double**
i	Индекс элементов массива	Int
j	Индекс элементов массива	Int
P	Количество строк	Int
L	Количество столбцов	Int
*locale	Локаль	char*
**S	Матрица 1	double**
**F	Матрица 2	double**
**R	Результат применения функции	double**

5/a)

```
#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <locale.h>

#include <malloc.h>

double* v(double**A, double*B, int n,int k){

double *C;

C=(double *) malloc(n*sizeof(double));

for (int i=0;i<n;i++)

{

for (int j=0;j<k;j++)

{

C[i]+=A[i][j]*B[j];

}

}

return(C);

}
```

```

int main(){
int P;
int L;
    char *locale=setlocale(LC_ALL, "");
double **S;
double *F;
scanf ("%d",&P);
scanf("%d",&L);
S=(double **) malloc (P * sizeof(double *));
    for (int i=0;i<P;i++)
    {
        S[i]=(double *) malloc(L*sizeof(double));
    }
F=(double *)malloc(P*sizeof(double));
for (int i=0; i<P; i++)
{
    for (int j=0; j<P;j++)
    {
        scanf("%lf",&S[i][j]);
    }
}
for (int i=0;i<P;i++)
{
    scanf("%lf",&F[i]);
}
double *R;
R=v(S,F,P,L);
for (int i=0;i<P;i++)
{

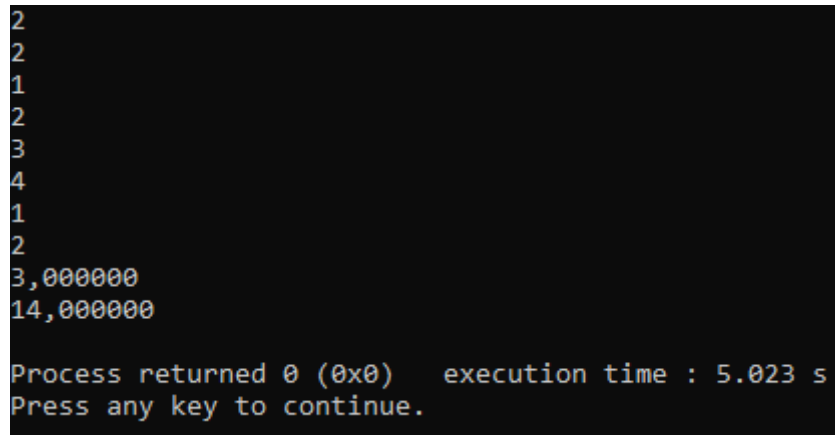
```

```

    printf("%lf\n",R[i]);
}
return 0;
}

```

6/a)



```

2
2
1
2
3
4
1
2
3
14
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
Process returned 0 (0x0)  execution time : 5.023 s
Press any key to continue.

```

5/b)

```

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <locale.h>
#include <malloc.h>

double** v(double**A, double **B, int n,int k){
double **C;
C=(double **) malloc (n * sizeof(double *));
    for (int i=0;i<n;i++)
    {
        C[i]=(double *) malloc(k*sizeof(double));
    }

for(int i = 0; i < n; ++i) {

```

```

    for(int j = 0; j < k; ++j) {
        C[i][j] = 0;
        for(int l = 0; l < n; ++l)
            C[i][j] += A[i][l] * B[l][j];
    }
}
return(C);
}

```

```

int main(){
    int P;
    int L;
    char *locale=setlocale(LC_ALL, "");
    double **S;
    double **F;
    scanf ("%d",&P);
    scanf("%d",&L);
    S=(double **) malloc (P * sizeof(double *));
    for (int i=0;i<P;i++)
    {
        S[i]=(double *) malloc(L*sizeof(double));
    }
    F=(double **) malloc (P * sizeof(double *));
    for (int i=0;i<P;i++)
    {
        F[i]=(double *) malloc(L*sizeof(double));
    };
    for (int i=0; i<P; i++)
    {
        for (int j=0; j<P;j++)

```

```

    {
        scanf("%lf",&S[i][j]);
    }
}
for (int i=0; i<P; i++)
{
    for (int j=0; j<P;j++)
    {
        scanf("%lf",&F[i][j]);
    }
}
double **R;
R=v(S,F,P,L);
for (int i=0; i<P; i++)
{
    for (int j=0; j<P;j++)
    {
        printf("%lf\n",R[i][j]);
    }
}
return 0;
}

```

6/b)

```
2
2
1
2
3
4
1
2
3
4
7,000000
10,000000
15,000000
22,000000

Process returned 0 (0x0)   execution time : 8.098 s
Press any key to continue.
```