Automata 2019 - PDF

Test Summary

- No. of Sections: 1
- No. of Questions: 10
- Total Duration: 100 min

Section 1 - Automata

Section Summary

- No. of Questions: 10
- Duration: 100 min

Additional Instructions:
None

Q1. You are given an integer N, print N+1 lines in the following manner

Case 1: If N=3, then the pattern would be –
3 3 3
3 1 3
3 2 3
3 3 3

Case 2: If N=4, then the pattern would be –
4 4 4 4 4
4 4 1 4 4
4 4 2 4 4
4 4 3 4 4
4 4 4 4 4

Testcase 1:

Input:
3
Excepted Return Value:
333
313
323
333

Sample Input

```
3
```

Sample Output

```
333
313
323
333
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. **Pattern**
You are given an integer N and a start value start, print 2*N lines in the following manner
If N= 4 and start = 3, then the pattern would be:
3
44
555
6666
6666
555
44
3
The input to the method IncrementPatternPrint of class IncrementPattern shall consist of a positive integer start value start and an integer N (Assume 0 < N < 100).
Do not return anything from the method. Print the required pattern
Each line of the output shall consist of 'numerals' only. There should be no spaces.

Input Format

Input contains two integers n and s

Output Format

Print the required format

Constraints

1<=n,s<=100

Sample Input

```
3  5
```

Sample Output

```
5
66
777
777
```

Sample Input

```
2  5
```

Sample Output

```
5
66
66
5
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. **Remove Vowels**

Given a string str, write a program to eliminate all the vowels from it.

The list of vowels In the English alphabet is : {a,e,i,o,u,A,E,I,0.U}

The Input to the function eliminateVowelString shall consist of a string str (containing only English letters) and returns a pointer to a string which does not contain vowels.

Example:
Input =”abcdefghijklmnopqrstuvwxyz” I
0utput="bcdfghjklmnpqrstvwxyz”

Useful Commands:

Strlen() is used to calculate the length of the string. The statement -int len = strlen(str);
Returns the length of the string str

Input Format

Input contains the string

Output Format

print the altered string

Constraints

1<= string_length<=1000

Sample Input

```
gAztkTJkCcmUVphMtGEDcWMMLSccLPvrMyLKTYYhkCYfZAiTDJKuS
```

Sample Output

```
gztkTJkCcmVphMtGDcWMMLSccLPvrMyLKTYYhkCYfZTDJKSfSwnnt
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. **A company is transmitting data to another server. The data is in the form of numbers. To secure the data during transmission, they plan to obtain a security key that will be sent along with the data. The security key is identified as the count of the repeating digits in the data.**
Write an algorithm to find the security key for the data.

Sample Input

```
1234234345
```

Sample Output

```
3
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. **A company provides network encryption for secure data transfer. The data string is encrypted prior to transmission and gets decrypted at the receiving end. But due to some technical error, the encrypted data is lost and the received string is different from the original string by 1 character. Arnold, a network administrator, is tasked with finding the character that got lost in the network so that the bug does not harm other data that is being transferred through the network.**

Write an algorithm to help Arnold find the character that was missing at the receiving end but present at the sending end.

Sample Input

```
abcdefghij abcdefghi
```

Sample Output

```
j
```

Sample Input

```
aaaabaaaa aaaaaaaa
```

Sample Output

```
b
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q6. **An e-commerce website wishes to find the lucky customer who will be eligible for full value cash back. For this purpose, a number N is fed to the system. It will return another number that is calculated by an algorithm. In the algorithm, a sequence is generated, in which each number is the sum of the two preceding numbers. Initially the sequence will have two 1's in it. The system will return the Nth number from the generated sequence which is treated as the order ID. The lucky customer will be the one who has placed that order.**
Write an algorithm to help the website find the lucky customer.

Sample Input

```
8
```

Sample Output

```
21
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q7. **In a science research lab, combining two nuclear chemicals produces a maximum energy that is the product of the energy of the two chemicals. The energy values of the chemicals can be negative or positive. The scientist wishes to calculate the sum of the maximized energies of the two elements when the reaction happens.**
Write an algorithm to find the total energy produced by the chemicals when they combine

Sample Input

```
6
-2 7 6 9 -3 -4
```

Sample Output

```
16
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q8. **Swap Value and index**
Given a unique positive integer array of length len with element ranging from 0 to (len -1),write a program to interchange the element value and its corresponding index values.
For example : if a[0]=3, a[1]=2,a[2]=0 and a[3]=1
Then output is : a[0]=2, a[1]=3, a[2]=1 and a[3]=0

The input to the function swapArr shall consist of an array arr its length len. The function should return an array after replacing the elements with their index values . the values in arr shall always be and cover all numbers between 0 to (length of array-1)

Useful Commands:
Malloc() is used to dynamically allocate memory in c. The statement- int*arr_arr=(int*)malloc(len*sizeof(int));
Creates an integer array arr_new of length len

Input Format

Input contains the array size and the values

Output Format

Print the altered array

Constraints

1<=array_size<=1000

Sample Input

Sample Output

```
9
6 4 2 3 5 0 1 7 8
```

```
5 6 2 3 1 4 0 7 8
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q9. **You are given an initial value as s and dimensions of the increment matrix as m and n.**
An increment matrix is the matrix whose elements are the incremented values of the initial

value s. N

For example -

if initial value s = 1 and dimesions are: m=3,n=3
Increment Matrix would be:

1 2 3
4 5 6
7 8 9

Multiply the original increment matrix with its transpose.

The input to the method transposeMultMatrix shall consist of
the initial value s and the dimensions of the increment matrix m and n (s, m and n all should be positive integers).

The method should return a 2-dimesional matrix for the multiplication matrix.

Sample Input

```
1 3 3
```

Sample Output

```
14 32 50
32 77 122
50 122 194
```

Sample Input

```
4 3 2
```

Sample Output

```
41 59 77
59 85 111
77 111 145
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q10. Pattern

Get input as N, square the N and generate the number from 1 to N2.

n= 4

```
 1*2*3*4
9*10*11*12
13*14*15*16
5*6*7*8
```

Input Format

Input contains n

Output Format

Print the pattern

Constraints

1<=n<=25

Sample Input

```
4
```

Sample Output

```
1*2*3*4
9*10*11*12
13*14*15*16
5*6*7*8
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - Automata

Q1

Test Case

Input

```
4
```

Output

```
44444
44144
44244
44344
```

Weightage - 20

Input

```
5
```

Output

```
55555
55155
55255
55355
```

Weightage - 20

Input

```
6
```

Output

```
6666666
6661666
6662666
6663666
```

Weightage - 20

Input

```
7
```

Output

```
7777777
7771777
7772777
7773777
```

Weightage - 20

Input

```
8
```

Output

```
88888888
88881888
88882888
88883888
```

Weightage - 20

Sample Input

```
3
```

Sample Output

```
333
313
323
333
```

Solution

```
#include<stdio.h>
int main()
{
    int n,counter=1,k;
    scanf("%d",&n);
    if(n%2==0){
```

```
            k=n+1;
        }
        else{
            k=n;
        }
        for(int i=0;i<n+1;i++){
            for(int j=0;j<k;j++){
                if(j==k/2 && i>=1 && i<=n){
                    printf("%d",counter++);
                }
                else{
                    printf("%d",n);
                }
            }
            printf("\n");
        }
    }
```

Q2

Test Case

Input

```
5  1
```

Output

```
1
22
333
4444
```

Weightage - 10

Input

```
10  2
```

Output

```
2
33
444
5555
```

Weightage - 10

Input

```
3  9
```

Output

```
9
1010
111111
111111
```

Weightage - 10

Input

```
5  5
```

Output

```
5
66
777
8888
```

Weightage - 10

Input

```
10  5
```

Output

```
5
66
777
8888
```

Weightage - 10

Input

```
6 5
```

Output

```
5
66
777
8888
```

Weightage - 10

Input

```
15 15
```

Output

```
15
1616
171717
18181818
```

Weightage - 10

Input

```
15  20
```

Output

```
20
2121
222222
23232323
```

Weightage - 10

Input

```
12 5
```

Output

```
5
66
777
8888
```

Weightage - 10

Input

```
12 12
```

Output

```
12
1313
141414
15151515
```

Weightage - 10

Sample Input

```
3 5
```

Sample Output

```
5
66
777
777
```

Sample Input

```
2 5
```

Sample Output

```
5
66
66
5
```

Solution

Header

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

```
#include <string.h>
#include <math.h>
#include <stdlib.h>
using namespace std;
 class IncrementPattern
{
    public:
    void IncrementPatternPrint(int n,int s);
};



void IncrementPattern::IncrementPatternPrint(int n , int s)
{
    int row,col,ctr;
    for(row=1;row<=n;row++,printf("\n"))
    {
        for(col=1;col<=row;col++)
            printf("%d",s);
            s++;
    }
    s--;
    for(row=n;row>=1;row--)
    {
        for(col=1;col<=row;col++)
            printf("%d",s);
            s--;
        printf("\n");
    }

}
```

Footer

```
int main()
{
    int n,s;
        scanf("%d %d",&n,&s);
        IncrementPattern ip;
        ip.IncrementPatternPrint(n,s);
    return 0;
}
```

Header

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
using namespace std;
 class IncrementPattern
{
    public:
    void IncrementPatternPrint(int n,int s);
};



void IncrementPattern::IncrementPatternPrint(int n , int s)
{
    int row,col,ctr;
    for(row=1;row<=n;row++,printf("\n"))
```

```
        {
            for(col=1;col<=row;col++)
                printf("%d",s);
                s++;
        }
        s--;
        for(row=n;row>=1;row--)
        {
            for(col=1;col<=row;col++)
                printf("%d",s);
                s--;
            printf("\n");
        }

    }
```

Footer

```
    int main()
    {
        int n,s;
            scanf("%d %d",&n,&s);
            IncrementPattern ip;
            ip.IncrementPatternPrint(n,s);
        return 0;
    }
```

Q3

Test Case

Input	Output
JUEyYAaXtrDKhuBaYWwNiFtcxKxhfHviPhGhXYGKhSekKckzpY	JyYXtrDKhBYWwNFtcxKxhfHvPhGhXYGKhSkKckzpY

Weightage - 5

Input	Output
gNHSeCDuJHRtVuBLvggSgLqLDgCZXZTjVFzBTVQBqphWtaShwU	gNHSCDJHRtVBLvggSgLqLDgCZXZTjVFzBTVQBqphWtShw

Weightage - 5

Input	Output
mTdxSkXRxqUziCXqNzUBPntZGtfRJDvJKryQAzycbEQFtJqWyuS	mTdxSkXRxqzCXqNzBPntZGtfRJDvJKryQzycbQFtJqWySHZpHhZ

Weightage - 10

Input	Output
fRNVwzHGCBAjwECLzqYEwBUUwPVKJbaNPAaPMJmgLwCTrHaLVTG	fRNVwzHGCBjwCLzqYwBwPVKJbNPPMJmgLwCTrHLVTGSGVCBJGTB

Weightage - 10

Input

```
ADmwJzCEAwLFqUVHmDxcnVxXwKvWHbcinDTYMzTBcwMEqcjzvNx
```

Output

```
DmwJzCwLFqVHmDxcnVxXwKvWHbcnDTYMzTBcwMqcjzvNxwrRhbG
```

Weightage - 10

Input

```
jcUqLLYdjRtGtRPBegywpiqJSkJWxxZinNdtKSigGyawkKkYQWa
```

Output

```
jcqLLYdjRtGtRPBgywpqJSkJWxxZnNdtKSgGywkKkYQWKQkRmNG
```

Weightage - 10

Input

```
SGHVWJjPwUUBUWJqzwQMJcFtCZrvgxfRbFmjvBChgLDawxtQVKd
```

Output

```
SGHVWJjPwBWJqzwQMJcFtCZrvgxfRbFmjvBChgLDwxtQVKdKTxh
```

Weightage - 10

Input

```
czYmKYgupUnbDugczyBFgjJQbfxRjkSbwyUjHJuLwvgCiKxpnmu
```

Output

```
czYmKYgpnbDgczyBFgjJQbfxRjkSbwyjHJLwvgCKxpnmCZXywKv
```

Weightage - 10

Input

```
uiEbvsEOeOoAEiEuUOEbEoaIttOeUaUEoUAbaEOsIAAUvEEoOou
```

Output

```
bvsbttbsvbbvvbstbbtvbstsbssstvvssvsbbvvtvbsvtsvbttv
```

Weightage - 10

Input

```
IsaaEssiOtIsAvAAEEOoiUteaiOuisbEEIAbOUOvvavbaoIisvs
```

Output

```
ssstsvtsbbvvvbsvsbbvtvbtvbbbtbvsvsbbvbtstsbstsbvbts
```

Weightage - 10

Input

```
ebuvuOoeAuIttooaouabtuEuOEsostbveiOAOAvEIiIIbtviOiO
```

Output

```
bvttbtsstbvvbtvssvbvvvttssststvtbststvttssbsbsstvvt
```

Weightage - 10

Sample Input

```
gAztkTJkCcmUVphMtGEDcWMMLSccLPvrMyLKTYYhkCYfZAiTDJK
```

Sample Output

```
gztkTJkCcmVphMtGDcWMMLSccLPvrMyLKTYYhkCYfZTDJKSfSwn
```

Solution

Header

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>
```

```
char* mystrchr(char* str, char ch)
{
    int index;
    for(index = 0 ; str[index] ; index++)
    {
        if(str[index] == ch)
            return str+index;
    }
    return NULL;

}
```

```
char * eliminateVowelString(char *str)
{
    int index = 0, update = 0;
   char*  ptr = NULL,vowels[] = "AEIOUaeiou";
   for(index = 0 ; str[index] ; index++)
   {
       ptr = mystrchr(vowels, str[index]);
        if(ptr == NULL)
            str[update++] = str[index];
   }
   str[update] = '\0';
   //printf("%s",str);
   return str;
}
```

Footer

```
int main()
{
    int test,len,ctr;
    char  str[1000];

        scanf("%s",str);
   eliminateVowelString(str);
   printf("%s",str);
       return 0;
}
```

Header

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>
```

```
char* mystrchr(char* str, char ch)
{
    int index;
    for(index = 0 ; str[index] ; index++)
    {
        if(str[index] == ch)
            return str+index;
    }
    return NULL;

}
```

```
char * eliminateVowelString(char *str)
{
    int index = 0, update = 0;
   char*  ptr = NULL,vowels[] = "AEIOUaeiou";
   for(index = 0 ; str[index] ; index++)
   {
       ptr = mystrchr(vowels, str[index]);
        if(ptr == NULL)
            str[update++] = str[index];
   }
   str[update] = '\0';
   //printf("%s",str);
   return str;
}
```

Footer

```
int main()
{
    int test,len,ctr;
    char  str[1000];

        scanf("%s",str);
   eliminateVowelString(str);
   printf("%s",str);
       return 0;
}
```

Q4

Test Case

Input **Output**

```
7643764376
```

```
4
```

Weightage - 25

Input **Output**

```
123456789
```

```
0
```

Weightage - 20

Input **Output**

```
12345321
```

```
3
```

Weightage - 20

Input **Output**

```
675322323
```

```
2
```

Weightage - 15

Input **Output**

```
1256761
```

```
2
```

Weightage - 20

Sample Input **Sample Output**

```
1234234345
```

```
3
```

Solution

```
#include<stdio.h>
int main()
{
    long long int n;
    scanf("%lld",&n);
    int arr[10]={0},rem;
    while(n!=0){
        rem=n%10;
        arr[rem]++;
        n=n/10;
    }
```

```
    int count=0;
    for(int i=0;i<10;i++){
        if(arr[i]>1){
            count++;
        }
    }
    printf("%d",count);
}
```

Q5

Test Case

Input

```
abcdefghijk adgjk
```

Output

```
bcefhi
```

Weightage - 10

Input

```
jvbdjfbvjhdfbjhvbfdjb b
```

Output

```
jvdjfbvjhdfbjhvbfdjb
```

Weightage - 30

Input

```
sixphrase-mySlate siphras-mSlae
```

Output

```
xeyt
```

Weightage - 20

Input

```
jkbvbbfbvbvjjbjbkjnkbnjknbdbdnbk nb
```

Output

```
jkbvbbfbvbvjjbjbkjknjknbdbdnbk
```

Weightage - 20

Input

```
jkbfj kbf
```

Output

```
jj
```

Weightage - 20

Sample Input

```
abcdefghij abcdefghi
```

Sample Output

```
j
```

Sample Input

Sample Output

aaaabaaaa aaaaaaaa	b

Solution

```
#include<stdio.h>
#include<string.h>
void missingCharacter(char str1[],int len1,char str2[],int len2){
    int i=0,j=0;
    while(i<len1 && j<len2){
        if(str1[i]!=str2[j]){
            printf("%c",str1[i]);
            i+=1;
        }
        else{
            i+=1;
            j+=1;
        }
    }
    for(int k =i;k<len1;k++){
        printf("%c",str1[k]);
    }
    }
int main()
{
    char str1[1000],str2[1000];
    scanf("%s %s",str1,str2);
    int len1=strlen(str1);
    int len2=strlen(str2);
    missingCharacter(str1,len1,str2,len2);
}
```

Q6

Test Case

Input

2

Output

1

Weightage - 20

Input

3

Output

2

Weightage - 20

Input

10

Output

55

Weightage - 20

Input

```
12
```

Output

```
144
```

Weightage - 20

Input

```
9
```

Output

```
34
```

Weightage - 20

Sample Input

```
8
```

Sample Output

```
21
```

Solution

```
#include <stdio.h>
int luckyCustomer(int n)
{
    if (n <= 1)
        return n;
    return luckyCustomer(n - 1) + luckyCustomer(n - 2);
}

int main()
{
    int n;
    scanf("%d",&n);
    printf("%d", luckyCustomer(n));
    getchar();
    return 0;
}
```

Q7

Test Case

Input

```
5
10000000 10000000 -2 -3 -4
```

Output

```
20000000
```

Weightage - 25

Input

```
7
1000000000 -1 -2 -3 -4 -5 -6
```

Output

```
999999999
```

Weightage - 10

Input

```
5
1  2  3  4  -9
```

Output

```
7
```

Weightage - 25

Input

```
10
1  2  3  4  5  6  7  8  9  10
```

Output

```
19
```

Weightage - 20

Input

```
5
-1  -2  -3  -4  -5
```

Output

```
-6
```

Weightage - 20

Sample Input

```
6
-2  7  6  9  -3  -4
```

Sample Output

```
16
```

Solution

```
#include<stdio.h>
int main()
{
    long long int n;
    scanf("%lld",&n);
    long long int arr[n],index;
    long long int max=-9999999999;
long long int min=9999999999;
for(int i=0;i<n;i++){
    scanf("%lld",&arr[i]);
    if(arr[i]>max){
        max=arr[i];
        index=i;
    }
    if(arr[i]<min){
        min=arr[i];
    }
}
long long int secMax=-9999999999;
 for(int i=0;i<n;i++){
        if(arr[i]>secMax && i!=index){
            secMax=arr[i];
        }
 }
 if(max<0){
     printf("%d",max+min);
 }
 else{
     printf("%d",max+secMax);
```

```
        }

    }
```

Q8

Test Case

Input

```
18
6 9 2 11 17 5 1 8 3 16 4 14 7 15 0 10 12 13
```

Output

```
14 6 2 8 10 5 0 12 7 1 15 3 16 17 11 13 9 4
```

Weightage - 5

Input

```
12
6 7 10 0 9 11 2 1 8 4 3 5
```

Output

```
3 7 6 10 9 11 0 1 8 4 2 5
```

Weightage - 5

Input

```
94
59 63 18 91 13 87 38 65 84 41 1 42 93 78 77 81
```

Output

```
42 10 59 88 51 52 50 28 17 27 41 67 19 4 54 18
```

Weightage - 10

Input

```
832
785 528 144 740 640 684 286 344 688 744 743 235
```

Output

```
327 378 163 719 635 788 262 267 463 159 257 291
```

Weightage - 10

Input

```
408
209 33 212 75 57 109 204 264 286 372 45 329 192
```

Output

```
114 123 146 143 260 76 156 315 37 239 100 367 2
```

Weightage - 10

Input

```
166
25 41 42 37 105 21 99 76 93 13 95 48 78 17 47
```

Output

```
141 27 48 60 151 107 71 138 36 45 146 55 119 9
```

Weightage - 10

Input

Output

```
857
843 200 445 386 490 323 66 559 383 237 300 694
```

```
178 610 315 220 230 57 104 280 62 121 579 343 1
```

Weightage - 10

Input

Output

```
31
18 9 10 7 28 2 21 27 0 17 12 11 23 13 16 22 4
```

```
8 18 5 19 16 25 29 3 30 1 2 11 10 13 23 20 14
```

Weightage - 10

Input

Output

```
929
619 59 325 613 12 236 195 288 154 176 679 364 2
```

```
895 617 390 338 610 203 72 675 603 206 643 491
```

Weightage - 10

Input

Output

```
302
12 161 47 19 29 84 264 273 71 141 100 38 18 93
```

```
15 255 77 238 109 199 259 119 45 34 195 161 0 1
```

Weightage - 10

Input

Output

```
45
30 9 24 1 36 27 31 22 8 6 37 39 29 33 28 16 21
```

```
21 3 37 36 26 44 9 28 8 1 39 18 22 32 31 20 15
```

Weightage - 10

Sample Input

Sample Output

```
9
6 4 2 3 5 0 1 7 8
```

```
5 6 2 3 1 4 0 7 8
```

Solution

Header

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


int * swapArr( int * arr,int size)
{
 int *sub,ctr;
 sub=(int*)malloc(sizeof(int)*size);
  for(ctr = 0 ; ctr < size ; ctr++ )
  {
     sub[arr[ctr]] =ctr;
```

Header

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


int * swapArr( int * arr,int size)
{
 int *sub,ctr;
 sub=(int*)malloc(sizeof(int)*size);
  for(ctr = 0 ; ctr < size ; ctr++ )
  {
     sub[arr[ctr]] =ctr;
```

```
  }

  return sub;
}
```

```
  }

  return sub;
}
```

Footer

Footer

```
int main()
{
   int *arr,ctr,size;
    //  clrscr();
   scanf("%d",&size);
  arr=(int*)malloc(sizeof(int)*size);
   for( ctr =0 ; ctr< size ; ctr++)
      scanf("%d",&arr[ctr]);
 arr=swapArr(arr,size);
  for( ctr =0 ; ctr< size ; ctr++)
    printf("%d ",arr[ctr]);
    return 0;
}
```

```
int main()
{
   int *arr,ctr,size;
    //  clrscr();
   scanf("%d",&size);
  arr=(int*)malloc(sizeof(int)*size);
   for( ctr =0 ; ctr< size ; ctr++)
      scanf("%d",&arr[ctr]);
 arr=swapArr(arr,size);
  for( ctr =0 ; ctr< size ; ctr++)
    printf("%d ",arr[ctr]);
    return 0;
}
```

Q9

Test Case

Input

```
3  4  2
```

Output

```
25  39  53  67
39  61  83  105
53  83  113  143
67  105  143  181
```

Weightage - 10

Input

```
5  4  2
```

Output

```
61  83  105  127
83  113  143  173
105  143  181  219
127  173  219  265
```

Weightage - 10

Input

```
2  4  3
```

Output

```
29  56  83  110
56  110  164  218
83  164  245  326
110  218  326  434
```

Weightage - 10

Input

```
2  3  4
```

Output

```
54  110  166
110  230  350
166  350  534
```

Weightage - 10

Input

```
4  2  5
```

Output

```
190  340
340  615
```

Weightage - 10

Input

```
4 5 2
```

Output

```
41 59 77 95 113
59 85 111 137 163
77 111 145 179 213
95 137 179 221 263
```

Weightage - 10

Input

```
5 3 4
```

Output

```
174 278 382
278 446 614
382 614 846
```

Weightage - 10

Input

```
4 3 5
```

Output

```
190 340 490
340 615 890
490 890 1290
```

Weightage - 10

Input

```
5 4 4
```

Output

```
174 278 382 486
278 446 614 782
382 614 846 1078
486 782 1078 1374
```

Weightage - 10

Input

```
4 4 4
```

Output

```
126 214 302 390
214 366 518 670
302 518 734 950
390 670 950 1230
```

Weightage - 10

Sample Input

```
1 3 3
```

Sample Output

```
14 32 50
32 77 122
50 122 194
```

Sample Input

```
4 3 2
```

Sample Output

```
41 59 77
59 85 111
77 111 145
```

Solution

Header

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



int * transposeMultMatrix(int s,int m ,int n)
{
   int *oMat=NULL,*tMat=NULL,*mulMat=NULL,ctr,ctr1,ctr2,ans,size;
   oMat = (int *)malloc(sizeof(int) * m*n);
   tMat = (int *)malloc(sizeof(int) * m*n);

   mulMat = (int *)malloc(sizeof(int) * m*m);
   for( ctr =0 ; ctr < m ; ctr++ )
   {
      for( ctr1= 0 ; ctr1 < m ; ctr1++ )
        *(mulMat+ctr*m + ctr1)=0;
   }
   for( ctr =0 ; ctr < m ; ctr++ )
   {
      for( ctr1= 0 ; ctr1 < n ; ctr1++ )
        *(oMat+ctr*n+ctr1)=s++;
   }
   for( ctr =0 ; ctr < n ; ctr++ )
   {
     for( ctr1= 0 ; ctr1 < m ; ctr1++ )
       *(tMat+ctr*m+ctr1)=*(oMat+ctr1*n+ctr);
   }

   for( ctr = 0,ans=0;ctr < m ; ctr++ )
   {
       for( ctr1 =0 ; ctr1 < m ; ctr1++ )
       {
           for( ctr2 = 0;ctr2<n;ctr2++)
           ans += ((*(oMat+ctr*n+ctr2))* (*(tMat+ctr2*m+ctr1)));
            *(mulMat+ctr*m+ctr1) = ans;
           ans=0;
       }
   }
   return mulMat;
}
```



```
int main()
{
   int s,m,n,ctr,ctr1,size;
   int *ans=NULL;
   scanf("%d %d %d",&s,&m,&n);
   ans=(int *)malloc(sizeof(int) *m*m );
   ans=transposeMultMatrix(s,m,n);
   for( ctr = 0;ctr < m ; ctr++ )
   {
       for( ctr1 =0 ; ctr1 < m ; ctr1++ )
               printf("%d ",*(ans+ctr*m+ctr1));
    printf("\n");
   }
   return 0;
}
```

Header

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



int * transposeMultMatrix(int s,int m ,int n)
{
   int *oMat=NULL,*tMat=NULL,*mulMat=NULL,ctr,ctr1,ctr2,ans,size;
   oMat = (int *)malloc(sizeof(int) * m*n);
   tMat = (int *)malloc(sizeof(int) * m*n);

   mulMat = (int *)malloc(sizeof(int) * m*m);
   for( ctr =0 ; ctr < m ; ctr++ )
   {
      for( ctr1= 0 ; ctr1 < m ; ctr1++ )
        *(mulMat+ctr*m + ctr1)=0;
   }
   for( ctr =0 ; ctr < m ; ctr++ )
   {
      for( ctr1= 0 ; ctr1 < n ; ctr1++ )
        *(oMat+ctr*n+ctr1)=s++;
   }
   for( ctr =0 ; ctr < n ; ctr++ )
   {
      for( ctr1= 0 ; ctr1 < m ; ctr1++ )
        *(tMat+ctr*m+ctr1)=*(oMat+ctr1*n+ctr);
   }

   for( ctr = 0,ans=0;ctr < m ; ctr++ )
   {
       for( ctr1 =0 ; ctr1 < m ; ctr1++ )
       {
           for( ctr2 = 0;ctr2<n;ctr2++)
           ans += ((*(oMat+ctr*n+ctr2))* (*(tMat+ctr2*m+ctr1)));
            *(mulMat+ctr*m+ctr1) = ans;
           ans=0;
       }
   }
   return mulMat;
}
```



```
int main()
{
   int s,m,n,ctr,ctr1,size;
   int *ans=NULL;
   scanf("%d %d %d",&s,&m,&n);
   ans=(int *)malloc(sizeof(int) *m*m );
   ans=transposeMultMatrix(s,m,n);
   for( ctr = 0;ctr < m ; ctr++ )
   {
       for( ctr1 =0 ; ctr1 < m ; ctr1++ )
                printf("%d ",*(ans+ctr*m+ctr1));
    printf("\n");
   }
   return 0;
}
```

Q10 **Test Case**

Input

```
5
```

Output

```
1*2*3*4*5
11*12*13*14*15
21*22*23*24*25
16*17*18*19*20
```

Weightage - 5

Input

```
7
```

Output

```
1*2*3*4*5*6*7
15*16*17*18*19*20*21
29*30*31*32*33*34*35
43*44*45*46*47*48*49
```

Weightage - 5

Input

```
10
```

Output

```
1*2*3*4*5*6*7*8*9*10
21*22*23*24*25*26*27*28*29*30
41*42*43*44*45*46*47*48*49*50
61*62*63*64*65*66*67*68*69*70
```

Weightage - 10

Input

```
15
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15
31*32*33*34*35*36*37*38*39*40*41*42*43*44*45
61*62*63*64*65*66*67*68*69*70*71*72*73*74*75
```

Weightage - 10

Input

```
13
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12*13
27*28*29*30*31*32*33*34*35*36*37*38*39
53*54*55*56*57*58*59*60*61*62*63*64*65
```

Weightage - 10

Input

```
12
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12
25*26*27*28*29*30*31*32*33*34*35*36
49*50*51*52*53*54*55*56*57*58*59*60
73*74*75*76*77*78*79*80*81*82*83*84
```

Weightage - 10

Input

```
17
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*17
35*36*37*38*39*40*41*42*43*44*45*46*47*48*49*50*51
69*70*71*72*73*74*75*76*77*78*79*80*81*82*83*84*85
```

Weightage - 10

Input

```
20
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*17*18*19*20
```

```
41*42*43*44*45*46*47*48*49*50*51*52*53*54*55*56*57
81*82*83*84*85*86*87*88*89*90*91*92*93*94*95*96*97
```

Weightage - 10

Input

```
25
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*17*18*19*20
51*52*53*54*55*56*57*58*59*60*61*62*63*64*65*66*67
101*102*103*104*105*106*107*108*109*110*111*112*11
```

Weightage - 10

Input

```
23
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*17*18*19*20
47*48*49*50*51*52*53*54*55*56*57*58*59*60*61*62*63
93*94*95*96*97*98*99*100*101*102*103*104*105*106*1
```

Weightage - 10

Input

```
27
```

Output

```
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*17*18*19*20
55*56*57*58*59*60*61*62*63*64*65*66*67*68*69*70*71
109*110*111*112*113*114*115*116*117*118*119*120*12
```

Weightage - 10

Sample Input

```
4
```

Sample Output

```
1*2*3*4
9*10*11*12
13*14*15*16
5*6*7*8
```

Solution

```
#include<stdio.h>
int main()
{
    int N, row, col, num;
    scanf("%d",&N);
    for(row = 0; row < N ; row+=2, printf("\n"))
    {
        for(col = 0, num = row * N + 1; col < N-1; col++)
            printf("%d*", num++);
            printf("%d", num);
    }
for(row = N % 2 == 0 ? N -1: N-2; row > 0 ; row-=2, printf("\n"))
    {
        for(col = 0, num = row * N + 1; col < N-1; col++)
            printf("%d*", num++);
            printf("%d", num);
    }
}
```

```
#include<stdio.h>
int main()
{
    int N, row, col, num;
```

```
    int N, row, col, num;

    scanf("%d",&N);
    for(row = 0; row < N ; row+=2, printf("\n"))
    {
        for(col = 0, num = row * N + 1; col < N-1; col++)
            printf("%2d*", num++);
            printf("%2d", num);
    }
for(row = N % 2 == 0 ? N -1: N-2; row > 0 ; row-=2, printf("\n"))
    {
        for(col = 0, num = row * N + 1; col < N-1; col++)
            printf("%2d*", num++);
            printf("%2d", num);
    }
}
```