1.printcharecterpattern
Input: 5 Expected output:
a ab
abc
abcd
abcde

testcase 2:
input: 1
Output: A

Given Code: #include<stdio.h>
void printcharecterpattern(int num)
{ int i,j value=1; char ch = 'a';
char print = ch; for(int
i=1;i<=num;i++,printf("\n"))
{ ch=print;
for(j=1;j<=i;j++)
printf("%c",ch++);
}
}

int main()
{
int num;
scanf("%d",&num);
printcharecterpattern(num);

}
Corrected Code: #include<stdio.h>
void printcharecterpattern(int num)
{ int
i,j,value=1;
char ch = 'a';
char print = ch;
for(i=1;i<=num;i++,printf("\n"))
{ ch=print;
for(j=1;j<=i;j++)
printf("%c",ch++);
}
}

You are required to fix all logical errors in the given code. You can click on Compile, Run anytime to check the compilation/execution status of the program. You can use System.out.println to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required. Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods. The method printColor(intnum) of the class Color is supposed to print names of color according to given input numbers num When the values of num equal 1,2,3,4 the function prints "Red","Black","White","Green" respectively for any other values of num it should print "No color". The method complies fine but fails to return the desired results for some cases. Your task is to fix code so that it passes all the testcases

Given Code:

```
int printcolor(int num)
{
        switch (num)
        {
                case 1: printf("red"); case
                2: printf("black"); case 3:
                printf("white"); case 4:
                printf("green"); default:
                printf("no color"); break;
        }
  }
```

Corrected Code:

```
int printcolor(int num)
{
        switch (num)
        {
                case 1: printf("red");
                break; case 2:
                printf("black"); break;
                case 3: printf("white");
                break; case 4:
                printf("green"); break;
                default: printf("no color");
                break;
        }
  }
```

The method printpattern(int) of class drawpattern is expected to expected to print the first n (n > 0) Lines of the pattern

TESTCASES

TestCase 1

Input: 4

Expected Return value:

11

1111

111111

11111111 TestCase 2:

Input: 1

Expected Return Value:

11

Given Code:

```
int printpattern(int n)
{
        int i,j,print =1;
        for(i=1;i<=n;i++)
        for(j=1;j<=2 * i;j++)
        { printf("%d",print );
        }
        printf("\n");
}
```

Corrected Code:

```
int printpattern(int n)
{
        int i,j,print =1; for(i=1;i<=n;i++)
        { for(j=1;j<=2 * i;j++)
                { printf("%d",print );
                }
                printf("\n");
        }
}
```

4. Multiply the middle number with maximum of three numbers

TESTCASE 1
Input 5,7,4
Expected return value:
35

TESTCASE 2
Input
11,12,13
Expected return value:
156

Given Code:

```
#include<stdio.h>
int multiplynumber(int a,int b,int c)

{ int result,min,max,mid;
        max=(a>b)?((a>c)?a:c((b>c)?b:c);
        min=(a<b)?((a<c)?a:c((b<c)?b:c);
        mid=(a+b+c)-(min+max);
        result=(max*mid);
return result;  }
```

Corrected Code:

```
#include<stdio.h>
int multiplynumber(int a,int b,int c)
{       int     result,min,max,mid;     max=
        (a>b)?((a>c)?a:c):((b>c)?b:c);
                                        )
                                        :
                                        )
                                        :
        min=(a<b)?((a<c)?a:c):((b<c)?b:c);
        //max=  (a>b)  ?  ((a>c)?a:c    ((b>c)?b:c);
        //min=   (a<b)  ?  ((a<c)?a:c    ((b<c)?b:c);

        mid=(a+b+c)-(min+max);
        result=(max* mid); return
        result;
}
```

Question :5

You are required to fix all logical errors in the given code. You can click on Compile & Run anytime to check the compilation/execution status of the program. You can use System.out.println to debug your code. The submitted code should be logically/syntactically correct and pass all test cases. Do not write the main() function as it is not required. Code Approach: For this question, you will need to correct the given implementation We do not expect you to modify the approach or incorporate any additional library methods. The function sortArray(int * arr,intlen) accepts an integer array arr of length (len>0) as an input and perform an in place sort operation on it. The function is expected to return the input array sorted in descending order The function compiles successfully but fails to return the desired results due to logical errors Your task is to debug the program to pass all the test cases

TESTCASE 1:
Input:
[23, 12, 14, 24, 21], 5
Expected Return Value:
[24, 23, 21, 14, 12]

TESTCASE 2:
Input:
[1, 1, 1, 1, 1], 5
Expected Return Value:
[1, 1, 1, 1, 1]
Given Code:

```
int *sortArray(int *arr,int *len)
{ int i=0,j=0,temp=0,index=0;
        for(i=0;i<len;i++)
        { for(j=i+1;j<len;j++)
                { if(arr[i]>arr[j])
                        { temp=arr[i];
                                arr[i]=arr[j];
                                arr[j]=temp;
                        } return
                arr;
                }
        }
}
```

Corrected Code:

```
int *sortArray(int *arr, int len)
{ int i=0,j=0,temp=0,index=0;
        for(i=0;i<len;i++)
        { for(j=i+1;j<len;j++)
                { if(arr[i]<arr[j])
                        { temp=arr[i];
                                arr[i]=arr[j];
                                arr[j]=temp;
                        }
                }
        } return
        arr;

}
```

Question:6

You are required to complete the given code by reusing existing functions. click on the helper code tab to find out the details of functions/classes provided for reuse you can click on compile & run anytime to check the compilation /execution status of the program you can use system.out.println to debug your code The submitted code should be logically/syntactically correct and pass all testcase . . Do not write the main() function as it is not required.

Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods. The function countElement(int *arr, int len, int n) is supposed to return the numbers of elements in the inputs array arr of length len, which are greater than twice of the input number n The function looks fine but given a compilation error Your task is to fix the program so that it passes all the testcases

TESTCASE 1:
Input:
[-2, -4, -3, -5, -6, -7, -8], 7, 3
Expected Return Value: 0

TESTCASE 2:
Input:
[22, 55, 66, 33, 44, 77], 6,13 Expected Return Value:
5

PROGRAM:

Given Code:

```
int countElement(int arr, int len, int n) {
        inti,count=0; for(int
        i=0;i<len;i++)
                {
                if(arr[i]>2n)
                        { count=-
                        1;
                }
        } return
        count;
}
```

Corrected Code:

```
int countElement(int arr, int len, int n)
{ int i,count=0;
        for(i=0;i<len;i++)
        { if(arr[i]>2*n)
                {
                        //count=-1;       count+=1;
                }
        }
        return count;
}
```

Question:7

The method countdigit(int sum) of class digits is supposed to return the value remainder when the input arguments num(num>0) is divided by the number of digits in num.

Given Code:

```
countdigit(int sum)
 { int count=0;
        while(num)
        {
                num=num/10; count++;
        }
        return (num%count); }
```

Corrected Code:

```
countdigit(int
sum)
 {
        int count=0,safe;
        safe=num;
        while(num)
        {
                num=num/10; count++;
        } num=safe; return
        (num%count);
}
```

Question:8

The Function reverseArray(intarr[])of class sort Array arr of an arguments For example, if the input array arr is {20,30,10,40,50} the function is expected to return{50,40,10,30,20} The function compiles successfully but fails to return the desired result due to logical errors Given Code:

```
int arrayReverse(int *arr,int len)
{ int i,temp,originallen=len;
        for(i=0;i<=originallen/2;i++)
        { temp=arr[len-1];
                arr[len-1]=arr[i];
                arr[i]=temp;
                len+=1;
        } return
arr; }
```

Corrected Code:

```
int arrayReverse(int *arr,int len)
{ int i,temp,originallen=len;
        for(i=0;i<=originallen/2;i++)
        { temp=arr[len-1];
                arr[len-1]=arr[i];
                arr[i]=temp;
                len+=1; len-=1;
        } return
        arr;
}
```

Question:9
Given Code:

```
char checkGrade(int score)
{ if(score<=60)
                return „D‟;
        else if((61<=score)&&(score<=75)) return
                „C‟;
        else if((76<=score)&&(score<=90))
        return „B‟; else
                return „A‟;
} int
main()
{
        int score;
scanf(“%d”,&score);     printf(“%c”,
checkGrade(score));     return 0;
}
```

Corrected Code:

```
char checkGrade(int score)
{ if(score<=60)
                return „D‟;
        else if((61>=score)&&(score<=75)) return
                „C‟;
        else if((76>=score)&&(score<=90))
        return „B‟; else
                return „A‟;
} int
main()
{
        int score;
scanf(“%d”,&score);     printf(“%c”,
checkGrade(score));     return 0;
}
```

Question:10
The function findMaxElement(int *arr1,int len1,int *arr2,int len2) accepts two integer arrays arr1,arr2 of length len1,len2 respectively. It is supposed to return the largest element in both the input arrays. Another function sortArray(int *arr,intlen) sorts the input array arr of length len in ascending order and returns the sorted array.
Your task is to use sortArray(int *arr,intlen) function and complete the code in findMaxElement(int *arr1,int len1,int *arr2,int len2) so that it passes all test cases.

TESTCASE 1:
Input:
[2, 5, 1, 3, 9, 8, 4, 6, 5, 2, 3, 11], 12, [11, 13,
2, 4, 15, 17, 67, 44, 2, 100, 0, 23]11
Expected Return Value: 100 TESTCASE
2:

Input:
[100, 22, 43, 912, 56, 89, 85], 7, [234, 123, 456, 234, 890, 101], 6 Expected
Return Value:
912

Given Code:

```
int *sortArray(int *arr,int *len)
{ int i=0,j=0,temp=0,index=0;
        for(i=0;i<len;i++)
        {
            for(j=i+1;j<len;j++)
            {
                if(arr[i]>arr[j])
                {
                        temp=arr[i]; arr[i]=arr[j];
                        arr[j]=temp;
                }
            }
        }
        return arr;
}

findMaxElement(int *arr1,int len1,int *arr2,int len2)
{
        //WRITE DOWN CODE HERE
        arr1= sortArray(arr1,len1);
        arr2= sortArray(arr2,len2);
        if(arr1[len1-1]>arr2[len2-1])
                return arr1[len1-1];
        else
                return arr2[len2-1];

}
```

Question 11:

The function getarraysum(int * arr, int len)is supported to calculate and return the sum of elements of the input array arr of length len(len>0) The function compiles successfully but fails to return the desired result due to logical errors.

Given Code:

```
int getarraysum(int *arr, int len)
{ int sum = 0;
        for(i=0;i<len;i-i-1)
        { sum = arr[i];
        }
        return sum;
}
```

Corrected Code:

```
int getarraysum(int *arr, int len)
{ int sum = 0;
        for(i=0;i<len;i++)
        { sum+ = arr[i];
        }
        return sum;
}
```

Question:12

The methods GetDigitSum(intarr[]) of class DigitSum accepts an integers array arr it is supposed to calculate the sum of digits of the even of the smallest elements in the input array it returns 1 if the calculated sum is even and returns 0 otherwise However there is a compliation error in the code your task is to fix it so that the program works for all the input values Note The methods getdigitSum uses another method getSum(int sum) which returns the sum of the digits of the input number num

Given Code:

```
int getDigitSum(int arr[])
{ int result,len=arr.length; for(int
        i=0;min=arr[0];i<len;i++)
        { if(arr[i]<min)
                min=arr[i];
        }
        results=getSum(min)
        if(results%2==0)
                return 1; else
        min==arr[j];
}
int getSum(int num)
{
        //WRITE YOUR CODE HERE
}
```

Corrected Code:

```
int getDigitSum(int arr[])
{ int result,len=arr.length; int
        min;
        for(int i=0;min=arr[0];i<len;i++)
        { if(arr[i]<min)
                min=arr[i];
        }
        results=getSum(min) if(results%2==0)
                return 1;
        else
                return 0;
}
int getSum(int num)
{
        //WRITE YOUR CODE HERE
        int rem,sum=0;
while(num)
        {
                rem=num%10;
sum+=rem;
                num/=10;
        }
        return sum;
}
```

Question:13
Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.
Lisa always forgets her birthday which is on 5Th july
In order to help her we have function CheckBirthDay(char *month,int day) which takes day and month as inputs and returns 1 if its her birthday and returns a 0 otherwise The function compiles fine but to return desired results for some cases Your task to fix the code so but that it passes at test cases
15(1)

TestCase 1;
Input
July 13
Expected Return Value: 0

TestCase 2:
Input
April 3
Expected Return Value:
0

Given Code:

```
int checkBirthday(char* month,int day)
{ if(strcmp(month,"july")  ||   (day =5)) return
        1;
        else return 0;
} int
main()
{ char inp[]="july"; int day=5;
        if(checkBirthday(inp,day)==1)
        printf("Yes");
        else printf("No");
        return 0 ;
}
```

Corrected Code:

```
int checkBirthday(char* month,int day)
{ if(strcmp(month,"july")==0 &&   (day ==5)) return
        1;
        else return 0;
} int
main()
{ char inp[]="july"; int day=5;
        if(checkBirthday(inp,day)==1)
        printf("Yes");
        else printf("No");
        return 0 ;
}
```

Question:14

Matrix Adding odd diagonal elements int calculateMatrixSum(int m, int n, int mat[m][n])

```
{
        //WRITE YOUR CODE HERE
        int i,j,sum=0,row=m,col=n;
        if(row>0 && col>0)
        {
                for(i=0;i<row;i++)
                {
                        for(j=0;j<col;j++)
                        {
                                if(i==j)
                                {
                                        if(mat[i][j]%2==0)
                                                sum+=mat[i][j];
                                }
                        }

                }
        }
        return sum;
}
```

Question:15

Manchester Encoding

int Manchester(int *arr, int len)

```
{
        //WRITE YOUR CODE HERE
int i;
        int *res=(int *)malloc(sizeof(int)*len);
        res[0]=arr[0];  //res[0]=(arr[0]!=0);
for(i=1;i<len;i++)
res[i]=arr[i]^arr[i-1];
        return res;
}
```

Question:16

Matrix Sum

Given Code:

```
int MatrixSum(int m, int n, int mat[m][n])
{
        int i,j,sum=0;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        sum+  =mat[i]  (j);
                }
        }
        return sum;
}
```

Corrected Code:

```
int MatrixSum(int m, int n, int mat[m][n])
{
	int i,j,sum=0;
	for(i=0;i<m;i++)
	{
		for(j=0;j<n;j++)
		{
			sum+=mat[i][j];
		}
	}
	return sum;
}
```

Question 17:

Replace all the elements of the array with the maximum element of array.

```
//WRITE DOWN YOUR CODE HERE
#include<stdio.h>
int * maxReplace(int  *arr, int len)
{
	int i
if(len>0)
	{
		int max=arr[0];
for(i=1;i<len;i++)
		{
			if(max<arr[i])
				max=arr[i];
		}
		for(i=0;i<len;i++)
			arr[i]=max;

	}
	return arr;
}
```

Question 18:

Find the number of occurrences of a given value in the array.

Given Code:

```
#include<stdio.h>
int occurrence(int  *arr, int len,int value)
{
	int i=0,count=0;
while(i<len)
	{
		if(arr[i]==value)
	count++;

	}
	return count;
}
```

Corrected Code:
```
#include<stdio.h>
int occurrence(int  *arr, int len,int value)
{
        int i=0,count=0;
        while(i<len)
        {
                if(arr[i]==value)
                count++;
i++;
        }
        return count;
}
```

Question 19:

The function patternPrint(int n) supposed to print n number of lines in the following pattern For n=4 the pattern should be:

1
1 1
1 1 1
1 1 1 1

The function complies successfully but fails to return the desired results due to logical errors
Your task is to debug the program to pass all the test cases

Given Code:
```
void patternPrint(int n)
{ int print=1,i,j; for(int
        i=0;i<n;i++)
        { for(j=0;j<=i;j++)
                        {
                        printf("%d",print);
                }
            } print("\n");
        }
}
```

Corrected Code:
```
void patternPrint(int n)
{
        int print=1,i,j; for(i=0;i<n;i++)
        { for(j=0;j<=i;j++)
                        {
                        printf("%d",print);
                 } print("\n");
        }
}
```

Question:20

The function removeElement(int *arr,intlen,int index)takes an array arr of length len as an input. It is supposed to return an array len-1 after removing the integer at the given index in the input arrayarr. If the given index is out of bounds, then this function should return the input array arr. The function compiles successfully but fails to return the desired result due to logical errors */

WRITE YOUR CODE

```
int* removeelement( int *arr, int len, int index)
{
        int i,j;
        if(index<len)
        {
for(i=index;i<len-1;i++)
          {
                    arr[i]=arr[i+1];
          }       int *rarr
        =(int*)malloc(sizeof(int)*(len-1));
        for(i=0;i<len-1;i++)
                rarr[i]=arr[i];
          return rarr;
    } else   return arr; }
```

Question:21

Replace a given array with zeros and ones depending on the even or odd criteria of the array length. //WRITE DOWN YOUR CODE HERE

```
int *replaceValues(int *arr, int len)
{
        int i;
        for(i=0;i<len;i++)
                arr[i]=len%2;
        return arr;
}
```

Question: 22

Selection Sort Given Code:

```
int * sortArray(int *arr, int len)
{ int x=0,y=0,n=len; int
        index_of_min, temp;
        for(x=0;x<n;x++)
        {
                index_of_min=x;
                for(y=x;y<n;y++)
                {
                        if(arr[index_of_min]>arr[y])
                        {
                                index_of_min=y;
                        }
                }
                temp = arr[x]; arr[x] =
                arr[index_of_min];
                arr[index_of_min] = temp;

        } return
        arr;
}
```

Corrected Code:

```
int * sortArray(int *arr, int len)
{ int x=0,y=0,n=len; int
        index_of_min, temp;
        for(x=0;x<n;x++)
        {
                index_of_min=x;
                for(y=x;y<n;y++)
                {
                        if(arr[index_of_min]>arr[x])
                        {
                                index_of_min=y;
                        }
                }
                temp = arr[x]; arr[x] =
                arr[index_of_min];
                                        arr[index_of_min] = temp;

        } return
        arr;
}
```

QUESTION:23
Return the difference between two given times in seconds

TESTCASE

TestCase1:
Input:
Time:1:58:42, Time:2:1:45 Expected
Return values:
183

Testcase 2 Input:
Time:3:49:57, Time:2:45:57
Expected Return Values
3600

```
#include<stdio.h> struct
Time
{ int h; int m;
        int s;
};
typedef struct Time TIME;
toSeconds(TIME * gt)
{ int in_seconds;
        in_seconds = gt->h * 3600 + gt->m * 60 + gt->s; return
        in_seconds;
} int abs(int
val)
{ if (val< 0)  return -val;
        else
                return val;
}
```

```
diff_in_times(TIME *t1, TIME *t2)
{
        //WRITE DOWN YOUR CODE HERE
        int  t5,t6,res,result;
        t5=  toSeconds(t1);
        t6=  toSeconds(t2);
        res=          t5-t6;
        result=abs(res);
        return  result;
} int
main()
{
        TIME t1 = {1,58,42},  t2 = {2,59,45}; printf("%d",
        diff_in_times(&t1, &t2));
        return 0;
}
```

Question:24
Print the following Pattern
1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1

```
//WRITE DOWN YOUR CODE HERE
void printPattern(int n)
{
        int i,j;
        for(i=1;i<=n;i++,printf("\n"))
        {
                for(j=1;j<=i;j++)
                {
                        printf("%d",j);
                }
                for(j--;j>=1;j--)
                {
                        printf("%d",j);
                }
        }
}
```

Or

```
void printPattern(int n)
{
        int i,j,num=1;
        for(i=1;i<=n;i++)
        {
                num=num*10+1;
                printf("%d\n", num*num);
        }
}
```

Question:25

You are required to fix all logical errors in the given code. You can click on Compile & Run anytime to check the compilation/execution status of the program. You can use System.out.println to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required. Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The method deleteDuplicate(intarr[]) of classDistinctArray takes an array as an input it is supposed to remove duplicates integers from the input array arr such that for each distinct integer the first occurrence is retained and all the duplicates elements following it are removed for Example given input array
(2,3,2,2,5,6,6,7)
the expected output is (2,3,5,6,7)

The function complies successfully but fails to return the desired results due to logical errors Your task is debug the program to pass all the test cases Given Code:

```
int* deleteDuplicate (int *arr, int len)
{ int count=0,p,i,j,k=0,originalLength=len;
        for(i=0;i<len;i++)
        { for(j=i+1;j<len;j++)
                { if(arr[j]==arr[i])
                        { arr[k]=arr[k+1];
                        } len=len-1;
                        count=count+1;
                        j=i;
                }
        } return
arr; }
```

Corrected Code:

```
int* deleteDuplicate (int *arr, int len)
{ int count=0,p,i,j,k=0,originalLength=len;
        for(i=0;i<len;i++)
        { for(j=i+1;j<len;j++)
                { if(arr[j]==arr[i])
                        {
                                for(k=j;k<len-1;k++)
                                arr[k]=arr[k+1];
                        } len=len-1;
                        count=count+1;
                        j=i;
                }
        } return
        arr;
}
```

QUESTION:26

The function sameelementcount(int *arr,intlen)accepts an integer array arr of length len as a input and returns the number of elements in an arr which are even numbers and equal to the element to its right

```
//WRITE DOWN YOUR CODE HERE
int sameelementcount(int *arr, int len)
{ int  i,count=0;
        for(i=0;i<len-1;i++)
        { if((arr[i]%2==0)&&(arr[i]==arr[i++]))
                count++;
        }
        return  count;
}
```

QUESTION:27

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

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

TESTCASE 1:

Input: "bacdefghijklmnopgrstu"

Expected Return Value: "bcdfghjklmnpqrst"

TESTCASE 2:

Input: "bacdcfgh"

Expected Return Value: "bcdlgh" char

```
* removeVowel(char *str)
{
        int trav,,hold=0;
        for(trav=0;str[trav]!='\0';trav++)
        {
  if(str[trav]=='a'|| str[trav]=='e'|| str[trav]=='i'|| str[trav]=='o'|| str[trav]=='u'||
str[trav]=='A'|| str[trav]=='E'|| str[trav]=='I'|| str[trav]=='O'|| str[trav]=='U')
                {

                }
                else
                {
                   str[hold]=str[trav];
    hold++;
                }
        }
        str[hold]='\0';
printf("%s",str);
        return 0;
}
```

QUESTION:28

Half sort Array:

```
#include<stdio.h>
#include<limits.h> int
main()
{ int arr[]={10,12,25,6,13,8,19};
        int index,size,max,maxpos,min,minpos,temp,scope; size=sizeof(arr)/sizeof(arr[0]);
        for(index= 0 ; index < size; printf("%2d ",arr[index++]));
if(index%2==0)
                {
                        min = INT_MAX;
                                for(index =1; index<size; index++)
                        {
                                        if(arr[index] < min)
                                {
                                        min = arr[index]; minpos=index;
                                }
                        } temp = arr[index];
                        arr[index] = arr[minpos];
                        arr[minpos]=temp;
                } else
                {
                        max = INT_MIN;
                                for(index = 0 ; index<size; index++)
                        { if(arr[index] > max)
                                { max = arr[index];
                                        maxpos=index;
                                }
                        }
                        temp = arr[index]; arr[index]
                        = arr[maxpos];
                        arr[maxpos]=temp;
                }
        for(printf("\n"),index= 0 ; index < size;printf("%2d ",arr[index++])); return
        0;
}
```

Question : 30 WRITE YOUR CODE
Pyramid of alphabets

```
    a
   bcd
  efghi
jklmnop
```

```
#include<stdio.h> void
printPattern(int n)
{
        int i,j;
char ch='a';
for(i=1;i<=n;i++)
        {
          for(j=1;j<=n-i;j++)
printf(" ");          for(j=1;j<=2*i-
1;j++)        printf("%c",ch++);
  printf("\n");
        }
}
```

QUESTION:31
You have to encrypt a non-empty string phrase. The encryption adds a 'cyclic shift' to each letter where the value of this 'cyclic shift' is decided by the position of the letter from the end of its word. The shift value for each letter of a word is its index value (starting from 0) from the right-most character of the word.

EXAMPLE:
The shift values in 'yum feed' will be
yum: m->0, u->1, y->2 feed: d->0, e->1, e->2, f->3 which gives the
encryption avmigfd

Here, adding the shift with value 0 to letter 'm' gives 'm' + 0 = m;
values 1 to 'u' gives 'u' + 1 = v and values 2 to 'y' gives 'y' + 2 = a and so on
Note that the shift wraps around on reaching the end of the alphabets, i.e., the shift values for 'y' as shown above is 'a'.

INPUT:
The input to the function/method consists of a string.

OUTPUT:
Return the encrypted string NOTE:
Assume that the input string contains single space separating set of words

```
#include<stdio.h>       char*
encryption(char*   str);   int
main()
{ char str[]="zebra tiger";
        printf("%s",encryption(str));
        return 0;
}
char* encryption(char* str)
{
```

```
        //your CODE
        int len,index,value;
for(len=0;str[len]!="\0";len++)
        {
                if(str[index]="\0")
                {
                        value=0;
        continue;
                }
                if(str[index]+value<=122)
str[index]=str[index]+value++;          else
                        str[index]=str[index]+value++ -26;
        }
        return str;
}
```

QUESTION:32

The LeastRecentlyUsed(LRU) cache algorithm exists the element from the cache(when it's full) that was leastrecentlyused. After an element is requested from the cache, it should be added to the cache(if not already there) and considered the most recently used element in the cache. Initially, the cache is empty. The input to the function LruCountMiss shall consist of an integer max_cache_size, an array pages and its length len. The function should return an integer for the number of cache misses using the LRU cache algorithm. Assume that the array pages always has pages numbered from 1 to 50.

TEST CASE1:
Input: 3 16 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0
Expected Return Value: 11

TESTCASE 2:
Input:2 9 2 3 1 3 2 1 4 3 2
Expected Return Value: 8

```
#include <stdio.h>  int
main()
{
   int max_cache_size,pages_len,i,j,cache[100],pages[100],k,pageincache=0,misscount=0;

   scanf("%d %d",&max_cache_size,&pages_len);
   for(i=0;i< max_cache_size;i++)
          cache[i]=-1;

   for(i=0;i<pages_len;i++)
   {
      pageincache=0;
      scanf("%d",&pages[i]);

      for(j=0;j<max_cache_size;j++)
      {
         if(pages[i]==cache[j])
         {
            pageincache=1;
            for(k=j;k<max_cache_size;k++)
```

```
                {
                 cache[k]=cache[k+1];
                }
                cache[max_cache_size-1]=pages[i];
            }
        }
        if(pageincache==0)
        {
            misscount++;
            for(k=0;k<max_cache_size;k++)
            {
                cache[k]=cache[k+1];
            }
            cache[max_cache_size-1]=pages[i];
        }
    }
    printf("%d",misscount);
}
```