**1. Duplicate Elements**

Tim is working as a data entry staff in a college. His manager wants him to delete the duplicate student id from the entry. Help Tim in writing a program to delete the duplicate elements.

Include a function named **eliminateDuplicate** that accepts 2 arguments and its return type is void. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The output array is stored in a global variable named output1 and the number of elements in the output array is stored in the global variable named output 2.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer array.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

1

6

3

5

6

8

5

9

**Sample Output 1:**

1

6

3

5

8

9

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

void eliminateduplicate(int n,int a[]);

int op[20];

int k=0;

int main()

{

int c,n,a[20],i;

scanf("%d",&n);

if(n<0)

{

printf("Invalid input");

return 0;

}

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

{

scanf("%d",&a[i]);

if(a[i]<0)

{

printf("Invalid input");

return 0;

}

}

eliminateduplicate(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void eliminateduplicate(int n,int a[])

{

int j,i;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]==a[j])

{

a[j]=-1;

}

}

}

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

{

if(a[i]>0)

{

op[k]=a[i];

k++;

}

}

}

**2. Salary Calculation**

Jim got his salary. His salary calculations are as follows.

From his Basic amount he gets 50% of his basic for house Rent allowances and 75% of his basic as special allowances . If the number of days he worked is 31 he gets 500 extra. Write a program to calculate his gross salary after calculating all his salary split up.

Include a function named **calculateGross** that accepts 2 integer arguments and returns a float. The first integer corresponds to Jim's basic salary and the second integer corresponds to the number of days Jim has worked. The function returns a float that corresponds to the gross salary.

Print Invalid Input and terminate the program in the following cases:

1. Basic salary is greater than 10000

2. Number of working days is greater than 31

3. Basic salary is negative

4. Number of working days is 0 or negative

**Input and Output Format:**

Input consists of 2 integers. The first integer corresponds to Jim's basic salary and the second integer corresponds to the number of days he has worked.

Output consists of a single float that corresponds to Jim's gross salary. The gross salary is displayed correct to 2 decimal places.

**Sample Input 1:**

5000

30

**Sample Output 1:**

11250 .00

**Sample Input 2:**

5000  
0

**Sample Output 2:**

Invalid Input

#include<stdio.h>

float salary(int basic,int days);

int main()

{

int b,d;

float c;

scanf("%d",&b);

scanf("%d",&d);

if(b<0 || d<0)

{

printf("Invalid input");

return 0;

}

c=salary(b,d);

printf("%.2f",c);

return 0;

}

float salary(int basic,int days)

{

float h,r,sum;

if(days==30)

{

h=basic\*0.5;

r=basic\*0.75;

sum=basic+h+r;

}

if(days==31)

{

h=basic\*0.5;

r=basic\*0.75;

sum=basic+h+r+500;

}

return sum;

}

**3. Arithmetic Operation**

Write a program to perform a specific arithmetic operation

Include a function named **performArithmeticOperation** that accepts 3 integer arguments and returns an integer that corresponds to the result. The first and second arguments correspond to the input numbers and the third argument corresponds to the choice of arithmetic operation.

If argument 3 =1, calculate the sum of input1 and input2

If argument 3 =2, calculate the difference of input1 and input2

If argument 3 =3, calculate the product of input1 and input2

If argument 3 =4, calculate the quotient of input1 divided by input 2

If the first two argument's values is negative or greater than 32767 , the function returns -1.

If the third argument's value is not in the range 1 to 4, the function returns -1.

If the function returns -1, print Invalid Input.

**Input and Output Format:**

Input consists of 3 integers.

Output consists of an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

12

3

**Sample Output 1:**

48

**Sample Input 2:**

-67

2

1

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int operation(int n1,int n2,int n3);

int main()

{

int n1,n2,n3,c;

scanf("%d",&n1);

scanf("%d",&n2);

scanf("%d",&n3);

if(n1<0 ||n2<0||n3<0)

{

printf("Invalid input");

return 0;

}

c=operation(n1,n2,n3);

printf("%d",c);

return 0;

}

int operation(int n1,int n2,int n3)

{int d;

switch(n3)

{

case 1: d=n1+n2;

break;

case 2: d=n1-n2;

break;

case 3: d=n1\*n2;

break;

case 4: d=n1/n2;

break;

}

return d;

}

**4. searchKeys**

Read the question carefully and follow the input and output format.  
  
Given an integer array, first index represents the key & second index represents the value. Find keys for the given value.  
  
**Input and Output Format:**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. The next lineconsistts of an integer that represents the value to be searched.   
Output consist of an integer array.  
  
1) Print "Invalid array size" when size of the array is negative and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
3) Print "Key not found" when there is no keys found.  
  
Include a function named searchKeys(int array[], int size) whose return type is void.  
The output array is stored in a global variable named found.  
  
**Sample Input 1:**  
8  
1  
4  
2  
4  
3  
4  
5  
6  
4  
  
**Sample Output 1:**  
1  
2  
3  
  
**Sample Input 2:**  
5  
5  
6  
7  
8  
9  
-5  
  
**Sample Output 2:**  
Key not found

#include<stdio.h>

void searchkeys(int a[],int n);

int k;

int op[20];

int main()

{

int a[20],i,c,n;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

if(a[i]<0)

{

printf("key not found");

return 0;

}

}

searchkeys(a,n);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void serachkeys(int a[],int n)

{

int d,i;

k=0;

scanf("%d",&d);

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

{

if(a[i]==d)

{

op[k]=a[i-1];

k++;

}

}

}

**5. Minimum of 3**

Write a program to find the minimum of 3 numbers.

Include a function named **findSmallest** that accepts 3 integer arguments and returns an integer that corresponds to the minimum value.

**Input and Output Format:**

Input consists of 3 integers.

Output consists of an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

12

3

**Sample Output 1:**

3

#include<stdio.h>

int small(int n1,int n2,int n3);

int main()

{

int a,b,c;

scanf("%d",&a);

scanf("%d",&b);

scanf("%d",&c);

printf("%d",small(a,b,c));

return 0;

}

int small(int n1,int n2,int n3)

{

int min;

if(n1<n2&&n1<n3)

{

min=n1;

}

else if(n2<n3)

{

min=n2;

}

else

{

min=n3;

}

return min;

}

**6. sumPrimeArray**

Read the question carefully and follow the input and output format.  
  
John is working in a bank. He has created account details transaction in a file and protected it with a password. He sent the file to his manager for review. The file is protected with a password. The password is the sum of Prime numbers. Write a function to generate the password.  
  
**Input and Output Format:**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer, which is the sum.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
3) Print 0, when there are no prime numbers in a given input array.  
  
Include a function named sumPrimeArray(int array[], int size) whose return type is an integer, which is the prime sum.  
  
**Sample Input 1:**  
5  
1  
2  
3  
4  
5  
  
**Sample Output 1:**  
10  
  
**Sample Input 2:**  
3  
4  
8  
9  
  
**Sample Output 2:**  
0

#include<stdio.h>

int prime(int a[],int n);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",prime(a,n));

return 0;

}

int prime(int a[],int n)

{

int i,flag,sum=0,j;

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

{

flag=0;

if(a[i]==1)

{

flag++;

}

for(j=2;j<a[i];j++)

{

if(a[i]%j==0 )

{

flag++;

break;

}

}

if(flag==0)

{

sum=sum+a[i];

}

}

return sum;

}

**7. Sort and Insert**

Write a program to insert the given number in the appropriate place after sorting the input array in ascending order and store the sorted array in the output array.

Include a function named **sortAndInsert** that accepts 3 arguments and its return type is void. The first argument is the input array, the second argument is an int that corresponds to the size of the array and the third argument is the array element to be inserted. The output array is stored in a global variable named output1.

If the size of the array is negative or if any of the elements in the array are negative , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+2 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array. The last integer corresponds to the element to be inserted.

Output consists of an integer array.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

1

6

3

5

8

10

4

9

2

**Sample Output 1:**

1

2

3

4

5

6

8

9

10

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

#include<stdio.h>

void sort(int n,int a[],int b);

int k;

int op[20];

int main()

{

int n,a[20],b,c,i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

sort(n,a,b);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void sort(int n,int a[],int b)

{

int t,i,j;

scanf("%d",&b);

a[n]=b;

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

{

for(j=i+1;j<=n;j++)

{

if(a[i]>a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

k=0;

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

{

op[k]=a[i];

k++;

}

}

**8. Array Product**

Write a program to find the product of posive/nonnegative elements in a given array.

Include a function named **calculateProduct** that accepts 2 arguments and returns an int. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The function returns an int that corresponds to the product.

If the size of the array is negative or if it is greater than 10 or if any element in the array is more than 2 digits, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer that corresponds to the product of positive numbers in the array.

**Sample Input 1:**

8

1

-2

3

4

-6

8

10

-6

**Sample Output 1:**

960

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int product(int a[],int n);

int main()

{

int a[20],n,i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",product(a,n));

return 0;

}

int product(int a[],int n)

{

int i,pro=1;

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

{

if(a[i]>0)

{

pro=pro\*a[i];

}

}

return pro;

}

**9. findLargest**

Read the question carefully and follow the input and output format.  
  
Write a program to find the largest of the 3 given numbers.  
  
**Input and Output Format :**  
Input consists of 3 integers. Output consist of an integer that is the maximum.  
  
Print "Number too large" when any of given input numbers is greater than 32767 .  
Print "Number too small" when given input is a negative number.  
  
Include a function named findLargest(int num1, int num2, int num3) whose return type is an integer, which is the largest.  
  
**Sample Input 1:**  
2  
3  
4  
  
**Sample Output 1:**  
4  
  
**Sample Input 2:**  
98974  
  
**Sample Output 2:**  
Number too large  
  
**Sample Input 3:**  
-32767  
  
**Sample Output 3:**  
Number too small

#include<stdio.h>

int big(int n1,int n2,int n3);

int main()

{

int a,b,c;

scanf("%d",&a);

scanf("%d",&b);

scanf("%d",&c);

printf("%d",big(a,b,c));

return 0;

}

int big(int n1,int n2,int n3)

{

int max;

if(n1>n2&&n1>n3)

{

max=n1;

}

else if(n2>n3)

{

max=n2;

}

else

{

max=n3;

}

return max;

}

**10. Array Sorting**

Write a program to sort the first half of the input array elements in ascending order and the second half of the input array elements in descending order.

Include a function named **ascDescArray** that accepts 2 arguments and its return type is void. The first argument is the input array and the second argument is an int that corresponds to the size of the array.

If the size of the array is negative or if any element in the array is negative , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of n integers that correspond to the elements in the sorted array.

Assume that the maximum size of the array is 20.

**Sample Input 1:**

7

1

9

8

4

6

4

5

**Sample Output 1:**

1

4

8

9

6

5

4

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-5

**Sample Output 3:**

Invalid Input

#include<stdio.h>

void sort(int a[],int n);

int op[20];

int k;

int main()

{

int a[20],n,c,i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

sort(a,n);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void sort(int a[],int n)

{

int i,j,t;

for(i=0;i<n/2+1;i++)

{

for(j=i+1;j<n/2+1;j++)

{

if(a[i]>a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

k=0;

for(i=0;i<n/2+1;i++)

{

op[k]=a[i];

k++;

}

for(i=n/2+1;i<n;i++)

{

for(j=i+1;j<n;j++)

{

if(a[i]<a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

for(i=n/2+1;i<n;i++)

{

op[k]=a[i];

k++;

}

}

**11. Sum of Prime Cubes**

Given an input integer n, write a program to find the sum of the cubes of the prime numbers upto n. (including n)

Please note that 1 is neither prime nor composite.

Include a function named **sumCubeOfPrime** that accepts an integer argument and returns an integer. The function returns -1 if the input is a negative number or if it is greater than 3000.

If the function returns -1, print Invalid Input.

**Input and Output Format:**

Input consists of a single integer.

Output consists of a single integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

**Sample Output 1:**

160

**Sample Input 2:**

-241

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

50000

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int prime(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",prime(n));

return 0;

}

int prime(int n)

{

int i,sum=0,flag,j;

for(i=2;i<=n;i++)

{

for(j=2;j<=i-1;j++)

{

if(i%j==0)

{

break;

}

}

if(j==i)

{

sum=sum+(i\*i\*i);

}

}

return sum;

}

**12. maximumSum**

Read the question carefully and follow the input and output format.  
  
Given an Integer array, find out sum of Even and odd Numbers individually and find the maximum.  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of maximum of odd and even sum.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
  
Include a function named maximumSum(int numbers[], int size) whose return type is an integer,.  
  
**Sample Input 1:**  
5  
12  
13  
14  
15  
16  
  
**Sample Output 1:**  
42  
  
**Sample Input 2:**  
-13  
  
**Sample Output 2:**  
Invalid array size

#include<stdio.h>

int sum(int a[],int n);

int main()

{

int a[20],i,n;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",sum(a,n));

return 0;

}

int sum(int a[],int n)

{

int sum=0,odd=0,i;

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

{

if(a[i]%2==0)

{

sum=sum+a[i];

}

else

{

odd=odd+a[i];

}

}

if(sum>odd)

{

return sum;

}

else

return odd;

}

**13. Factorial**

Write a program to find the factorial of a given number.

Include a function named **findFactorial** that accepts an integer argument and returns an integer that corresponds to factorial. If the input value is negative or greater than 10, the function returns -1.

If the function returns -1, print Invalid Input.

**Input and Output Format:**

Input consists of a single integer.

Output consists of an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

**Sample Output 1:**

24

**Sample Input 2:**

-67

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int fact(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",fact(n));

return 0;

}

int fact(int n)

{

int i,fact=1;

for(i=1;i<=n;i++)

{

fact=fact\*i;

}

return fact;

}

**14. Sum of even numbers**

Given an integer n, write a program to find the sum of even numbers upto and including n.

Include a function named **sumEven**that accepts an integer argument and returns an integer that corresponds to the sum of the even numbers. The function returns -1 if the input is less than zero or if the roll number is greater than 32767.

If the function returns -1, print “Invalid Input”.

**Input and Output Format:**

The input consists of an integer.

The output consists of an integer that corresponds to the sum of the even numbers

**Sample Input 1:**

10

**Sample Ouput 1:**

30

**Sample Input 2:**

-8

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int even(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",even(n));

return 0;

}

int even(int n)

{

int i,sum=0;

for(i=1;i<=n;i++)

{

if(i%2==0)

{

sum=sum+i;

}

}

return sum;

}

**15. reverseNumber**

Read the question carefully and follow the input and output format.  
  
Write a program to find the reverse of a given input integer  
  
**Input and Output Format :**  
Input consists of an integer, n. Output consist of the reverse of the number n.  
  
Print "Number too large" when the given input number is greater than 32767  
Print "Number too small" when the given input numbers is a negative number.  
  
Include a function named reverseNumber(int num) whose return type is integer, the reverse of n.  
  
**Sample Input 1:**  
1234  
  
**Sample Output 1:**  
4321  
  
**Sample Input 2:**  
326357  
  
**Sample Output 2:**  
Number too large

#include<stdio.h>

int rev(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",rev(n));

return 0;

}

int rev(int n)

{

int d,r=0;

while(n>0)

{

d=n%10;

r=(r\*10)+d;

n=n/10;

}

return r;

}

**16. Savings Calculation**

Jim got salary for this month and he spends 50% of his salary for food and 20% of his salary for travel. If the number of days he worked is 31 he gets a bonus of Rs.500. Write a program to find how much he can save in his pocket after spending all these?

Include a function named **calculateSavings** that accepts 2 integer arguments and returns a float. The first integer corresponds to Jim's basic salary and the second integer corresponds to the number of days Jim has worked. The function returns a float that corresponds to the amount that Jim could save.

Print Invalid Input and terminate the program in the following cases:

1. Basic salary is greater than 9000

2. Number of working days is greater than 31

3. Basic salary is negative

4. Number of working days is negative

**Input and Output Format:**

Input consists of 2 integers. The first integer corresponds to Jim's basic salary and the second integer corresponds to the number of days he has worked.

Output consists of a single float that corresponds to Jim's savings. Jim's savings is displayed correct to 2 decimal places.

**Sample Input 1:**

7000

30

**Sample Output 1:**

2100 .00

**Sample Input 2:**

50000

**Sample Output 2:**

Invalid Input

#include<stdio.h>

float save(int s,int d);

int main()

{

int s,d;

float c;

scanf("%d",&s);

scanf("%d",&d);

c=save(s,d);

printf("%.2f",c);

return 0;

}

float save(int s,int d)

{

float f,t;

if(d==30)

{

f=s\*0.5;

t=s\*0.2;

s=s-(f+t);

}

if(d==31)

{

f=s\*0.5;

t=s\*0.2;

s=s-(f+t)+500;

}

return s;

}

**17. Perfect Number**

Write a program to find whether the given number is a perfect Number.

A number is a perfect number if the sum of the proper divisors of the number is equal to the number itself.

Include a function named **findPerfect** that accepts an integer argument and returns an integer. The function returns

1. 1 if the input is a Perfect Number

2. 0 if the input is not a Perfect Number

3. -1 if the input is a negative number or if it is greater than 32767

**Input and Output Format:**

Input consists of a single integer.

Output consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

6

**Sample Output 1:**

yes

**Sample Input 2:**

241

**Sample Output 2:**

no

**Sample Input 3:**

-9

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int perfect(int n);

int main()

{

int n;

int c;

scanf("%d",&n);

c=perfect(n);

if(c==1)

{

printf("yes");

}

else

printf("no");

return 0;

}

int perfect(int n)

{

int i,sum=0;

for(i=1;i<n;i++)

{

if(n%i==0)

{

sum=sum+i;

}

}

if(sum==n)

{

return 1;

}

else

return 2;

}

**18. nonWorkingDoctors**

Read the question carefully and follow the input and output format.  
  
A doctor survey results information is stored in 2 arrays. First array represents all doctors ids (working and non -working both). Second array represents only working doctor's id . Please find the doctor ids who are not working .  
  
**Input and Output Format :**  
First line of input corresponds to n1, the size of first array and next n1 lines correspond to the elements of the first array. The next line corresponds to n2, the size of second array and next n2 lines correspond to the elements of the second array  
Output is the id's of doctor who are not working  
  
Print "Invalid array size" when size of the array is a negative number and terminate the program  
Print "Invalid id" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named nonWorkingDoctors(int total[],int working[],int n,int m) whose return type is void.  
  
**Sample Input 1:**  
7  
7  
2  
3  
4  
5  
6  
1  
3  
3  
4  
5  
**Sample Output 1:**  
7  
2  
6  
1  
  
  
**Sample Input 2:**  
7  
7  
2  
3  
4  
5  
6  
-1  
**Sample Output 2:**  
Invalid id

#include<stdio.h>

void doctor(int n,int a[],int m,int b[]);

int op[20];

int k;

int main()

{

int n,m,a[20],b[20],i,j,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

scanf("%d",&m);

for(j=0;j<m;j++)

{

scanf("%d",&b[j]);

}

doctor(n,a,m,b);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void doctor(int n,int a[],int m,int b[])

{

int i,j;

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

{

for(j=0;j<m;j++)

{

if(a[i]==b[j])

{

a[i]=-1;

}

}

}

k=0;

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

{

if(a[i]>0)

{

op[k]=a[i];

k++;

}

}

}

**19. oddDigitsProduct**

Read the question carefully and follow the input and output format.  
  
Given an Integer array find out odd numbers and calculate the product of individual digits of each odd number.  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer array, multiplication of individual digit of each odd entry.  
  
1) Print "Invalid array size" when size of the array is negative and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named oddDigitsProduct(int array[ ], int size) whose return type is void.  
The output array is stored in a global variable named oddProducts.  
  
**Sample Input 1:**  
8  
12  
34  
45  
56  
22  
19  
21  
93  
**Sample Output 1:**  
20  
9  
2  
27  
  
**Sample Input 2:**  
4  
-3  
**Sample Output 2:**  
Invalid Input

#include<stdio.h>

void odd(int n,int a[]);

int op[20];

int k;

int main()

{

int n,c,i,a[20];

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

odd(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void odd(int n,int a[])

{

int d,pro,i;

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

{

pro=1;

if(a[i]%2!=0)

{

while(a[i]>0)

{

d=a[i]%10;

pro=pro\*d;

a[i]=a[i]/10;

}

k=k+0;

op[k]=pro;

k++;

}

}

}

**22. countNoOfConnections**

Read the question carefully and follow the input and output format.  
  
Given an input array, the elements are of format XYYY .where X represents the connection type. YYY represents the connection id.  
  
If X is 2 -> means 2G connection  
If X is 3 -> means 3G connection  
If X is 4 -> means 4G connection  
You need to find the number of type of connections.  
  
Note:  
If a particular connection type (starting with 2 or 3 or 4) is not available represent with zero in the corresponding position.  
  
Include a function named countNoOfConnections(int connection\_list[],int no) that returns the number of types of connections  
  
**Business Rules :**  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid connection" when there is any negative number available in the input array and terminate the program  
  
**Input and Output Format :**  
First line of input corresponds to n, next n lines corresponds to the elements of the array  
Output consists of the the number of type of connections. [1st line of the output corresponds to the number of 2G connections, 2nd line corresponds to the number of 3G connections and 3rd line corresponds to the number of 4G connections]  
  
**Sample Input 1 :**  
5  
2333  
3101  
2102  
4567  
3123  
  
**Sample Output 1:**  
2  
2  
1  
  
**Sample Input 2 :**  
2  
-2234  
  
  
**Sample Output 2:**  
Invalid connection

#include<stdio.h>

void conn(int a[],int n);

int c1,c2,c3;

int main()

{

int n,a[20],i,b,c,d;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

conn(a,n);

b=c1;

c=c2;

d=c3;

pirntf("%d\n",b);

pirntf("%d\n",c);

pirntf("%d",d);

return 0;

}

void conn(int a[],int n)

{

int i;

c1=0,c2=0,c3=0;

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

{

if(a[i]/1000==2)

{

c1++;

}

if(a[i]/1000==3)

{

c2++;

}

if(a[i]/1000==4)

{

c3++;

}

}

}

**23. Sum of Even Digits**

In the computer science department, HOD wants to divide students into two groups based on their roll numbers. He decided to find the sum of the even digits in the roll number of each student and decided to split them into 2 groups based on this. Write a program to find the sum of the even digits in a number.

Include a function named **addEvenDigits** to find the sum of even digits in a number. This function accepts an integer argument and returns an integer. The function returns -1 if the roll number is less than zero or if the roll number is greater than 32767. Refer function specifications given at the end of the problem for further details.

If the roll number is less than 0 or if it exceeds 32767, print “Invalid Input”.

**Input and Output Format:**

The input consists of an integer that corresponds to the roll number.

The output consists of an integer that corresponds to the sum of the even digits in the roll number.

**Sample Input 1:**

3487

**Sample Ouput 1:**

12

**Sample Input 2:**

-8

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int sum(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",sum(n));

return 0;

}

int sum(int n)

{

int d,sum=0;

while(n>0)

{

d=n%10;

if(d%2==0)

{

sum=sum+d;

}

n=n/10;

}

return sum;

}

**24. studentMarks**

Read the question carefully and follow the input and output format.  
  
The Given input is of the format XXXYY , where XXX is Id , YY is marks. Write a code to display the id and marks separately as given in the output formats. [Refer sample input and output]  
  
**Input and Output Format :**  
Input consists of a number. Refer sample output for output format.  
  
Print "Number too large" when any of given input numbers is greater than 32767 .  
Print "Number too small" when given input is a negative number.  
  
Include a function named studentMarks(int number) whose return type is void.  
  
**Sample Input 1:**  
12345  
  
**Sample Output 1:**  
123  
45  
  
**Sample Input 2:**  
-13  
  
**Sample Output 2:**  
Number too small

#include<stdio.h>

void marks(int n);

int d,m;

int main()

{

int n,c,b;

scanf("%d",&n);

marks(n);

c=d;

b=m;

printf("%d\n",c);

printf("%d",b);

return 0;

}

void marks(int n)

{

d=n/100;

m=n%100;

}

**26. nextPrime**

Read the question carefully and follow the input and output format.  
  
Write a program to find out the Next Prime to the given number.  
  
Hint: number is always less than 100.  
  
**Input and Output Format :**  
  
First line of input consists of n, the number. Output is a single integer that displays the next prime.  
  
Print "Number too large" when the given input number is greater than 32767 .  
Print "Number too small" when given input is a negative number.  
  
Include a function named nextPrime(int num) whose return type is an integer, the next prime.  
  
**Sample Input 1:**  
9  
  
**Sample Output 1:**  
11  
  
**Sample Input 2:**  
98987  
  
**Sample Output 2:**  
Number too large

#include<stdio.h>

int prime(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",prime(n));

return 0;

}

int prime(int n)

{

int c,i,j,flag;

for(i=n+1;i<100;i++)

{

flag=0;

for(j=2;j<i;j++)

{

if(i%j==0)

{

flag=1;

break;

}

}

if(flag==0)

{

c=i;

break;

}

}

return c;

}

**27. sumEvenIndex**

Read the question carefully and follow the input and output format.  
  
Write a program to find the sum of the indexes (positions) of even numbers in the Array. Consider 0 index as 1 and 1 index is 2 and so on……  
Note : Assume Array Index Starts From 1  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer, which is the sum.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named sumEvenIndex(int numbers[], int size) whose return type is an integer, which is the sum..  
**Sample Input 1:**  
7  
4  
2  
7  
9  
1  
10  
13  
  
**Sample Output 1:**  
9  
  
**Sample Input 2:**  
-13  
  
**Sample Output 2:**  
Invalid array size

#include<stdio.h>

int sum(int a[],int n);

int main()

{

int a[20],i,n;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",sum(a,n));

return 0;

}

int sum(int a[],int n)

{

int i,sum=0;

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

{

if(i==1 || i%2!=0)

{

sum=sum+i;

}

}

return sum;

}

**30. Repeated Salary Count**

John is working as a clerk in an organization where N number of people are working. His boss has asked him to get the count of employees who get same salary. Help him to get the count of repeated salary.

Include a function named **countRepeaters** that accepts 2 arguments and returns an int. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The function returns an int that corresponds to the number of repeaters.

If the size of the array is negative or if any of the array elements are negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer that corresponds to the number of repeaters.

Assume that utmost one element in the array would repeat.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

5

1000

2000

3500

2000

5000

**Sample Output 1:**

2

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

1000

-2000

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int maxcnt(int a[],int n);

int main()

{

int a[20],i,n;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",maxcnt(a,n));

return 0;

}

int maxcnt(int a[],int n)

{

int i,j,count=1;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]==a[j])

{

count++;

}

}

}

return count;

}

**31. Search Element**

Write a program to find whether a particular number appears in a given input array.

Include a function named **isElementPresent** that accepts 3 arguments and returns an int. The first argument is the input array, the second argument is an int that corresponds to the size of the array and the third argument is the element to be searched for. The function returns 1 if the search element is present in the array and returns 0 if the search element is not present in the array.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+2 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array. The last integer corresponds to the element whose count needs to be found.

Output consists of a string that is either 'yes' or 'no'.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

2

1

3

8

6

8

10

8

8

**Sample Output 1:**

yes

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int search(int n,int a[],int x);

int main()

{

int n,a[20],i,x,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

scanf("%d",&x);

c=search(n,a,x);

if(c==1)

{

printf("yes");

}

else

printf("no");

return 0;

}

int search(int n,int a[],int x)

{

int i,count=0;

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

{

if(a[i]==x)

{

count=count+1;

break;

}

}

if(count==1)

{

return 1;

}

else

return 2;

}

**32. occurenceDigit**

Read the question carefully and follow the input and output format.  
  
Given a number n, count the occurences of a number,x in n.  
  
**Input and Output Format :**  
The first line of the input consists of an integer n, the second line consists of an integer, which is the digit whoseoccurence is to be calculated. Output is an integer that gives the count.  
  
Print "Number too small" if any of the 2 input numbers is less than 0 and terminate the program.  
Print "Number too large" if any of the 2 input numbers is greater than 32767 and terminate the program.  
  
Include a function named occurenceDigit(int number,int digit) that returns an integer, which is the count of the digit.  
  
**Sample Input 1:**  
1122  
1  
**Sample Output 1:**  
2  
  
  
**Sample Input 2:**  
-2134  
**Sample Output 2:**  
Number too small

#include<stdio.h>

int occurence(int n,int d);

int main()

{

int n,d;

scanf("%d",&n);

scanf("%d",&d);

printf("%d",occurence(n,d));

return 0;

}

int occurence(int n,int d)

{

int count=0,d1;

while(n>0)

{

d1=n%10;

if(d1==d)

{

count++;

}

n=n/10;

}

return count;

}

**33. consecutiveDifference**

Read the question carefully and follow the input and output format.  
  
Given input as integer array in which consecutive elements should have a difference of 4 or more than 4.  
  
If the above condition matches display “1” else “0”  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer, which is the either 1 or 0  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named consecutiveDifference(int elements[], int size) whose return type is an integer either 1 or 0.  
  
**Sample Input 1:**  
6  
1  
5  
10  
6  
2  
7  
  
**Sample Output 1:**  
1  
  
**Sample Input 2:**  
-8  
  
**Sample Output 2:**  
Invalid array size

#include<stdio.h>

int cons(int n,int a[]);

int main()

{

int n,a[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

c=cons(n,a);

if(c>0)

{

printf("1");

}

else

printf("0");

return 0;

}

int cons(int n,int a[])

{

int count,i;

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

{

if(a[i]>a[i+1])

{

if(a[i]-a[i+1]>=4)

{

count++;

}

else

{

count=-1;

}

}

if(a[i+1]>a[i])

{

if(a[i+1]-a[i]>=4)

{

count++;

}

else

{

count=-1;

}

}

}

return count;

}

**34. Armstrong Number**

Write a program to find whether the given input number is an Armstrong Number.

Include a function named **checkArmstrong** that accepts an integer and returns an integer. The function returns

1. yes if the input is an Armstrong number

2. no if the input is not an Arnstrong number

3. Invalid Input if the input is a negative number or if the input is not a 3-digit number.

Print Invalid Input if the function returns -1.

**Input and Output Format:**

Input consists of a single integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

153

**Sample Output 1:**

yes

**Sample Input 2:**

161

**Sample Output 2:**

no

**Sample Input 3:**

2345

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int arm(int n);

int main()

{

int c,n;

scanf("%d",&n);

c=arm(n);

if(c>0)

{

printf("yes");

}

else

{

printf("no");

}

return 0;

}

int arm(int n)

{

int d,a,sum=0,count=0;

a=n;

while(n>0)

{

d=n%10;

sum=sum+(d\*d\*d);

n=n/10;

}

if(sum==a)

{

count=1;

}

else

{

count=-1;

}

return count;

}

**35. Sum of positive numbers in Array**

Write a program to find the sum of positive numbers in an array.

Include a function named**addPositives** that accepts 2 arguments and returns an int. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The function returns the sum of the positive numbers in the array.

If the size of the array is negative, print “Invalid Input” and terminate the program..

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Assume that the maximum size of the array is 20.

**Sample Input 1:**

5

3

5

-2

6

-6

**Sample Output 1:**

14

**Sample Input 2:**

**-5**

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int pos(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",pos(n,a));

return 0;

}

int pos(int n,int a[])

{

int i,sum=0;

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

{

if(a[i]>0)

{

sum=sum+a[i];

}

}

return sum;

}

**38. 5 Multiples --- Average**

Write a program to find the average of multiples of 5 upto 'n'. n is given as input.

Include a function named **findAverageBy5s** that accepts an integer argument and returns a float that corresponds to the average of multiples of 5.

If the input value is negative or greater than 32767, print Invalid Input and terminate the program.

**Input and Output Format:**

Input consists of a single integer.

Output consists of a floating point number. Output is displayed correct to 2 decimal places.

Refer sample output for formatting specifications.

**Sample Input 1:**

10

**Sample Output 1:**

7.50

**Sample Input 2:**

-67

**Sample Output 2:**

Invalid Input

#include<stdio.h>

float five(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%.2f",five(n));

return 0;

}

float five(int n)

{

int i;

float avg,sum=0,count=0;

for(i=1;i<=n;i++)

{

if(i%5==0)

{

sum=sum+i;

count++;

}

}

avg=sum/count;

return avg;

}

**39. Product of MaxMin Element**

Write a program to find the product of the maximum and minimum element in a given input array.

Include a function named **productOfMaxMin** that accepts 2 arguments and returns an int. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The function returns an int that corresponds to the product of maximum and minimum element.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer that corresponds to the product of maximum and minimum element in the array.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

2

12

3

4

6

8

10

9

**Sample Output 1:**

24

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int pro(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",pro(n,a));

return 0;

}

int pro(int n,int a[])

{

int i,p,t,j;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]>a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

p=a[0]\*a[n-1];

return p;

}

**40. University Type**

Write a program to find if the student is eligible for first, second or third grade universities by finding the average of their marks given in the input integer array.

Grade should be calculated as given below :

Average >80 First Grade University

Average >60 Second Grade University

Otherwise Third Grade University

Include a function named **calculateGrade** that accepts 2 arguments and returns an integer. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The function returns an integer that corresponds to the university type. The function returns 1 if the student is eligible for First Grade univesity, returns 2 if the student is eligible for Second Grade University, returns 3 if the student is eligible for Third Grade University and returns -1 if the average is greater than 99.

If the size of the array is negative or if any element in the array is negative or if the average marks scored by the student is greater than 99 , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a string --- “First Grade University” or “Second Grade University” or “Third Grade University” or “Invalid Input”.

Assume that the maximum size of the array is 20.

**Sample Input 1:**

5

92

87

78

74

80

**Sample Output 1:**

First Grade University

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-5

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int grade(int n,int a[]);

int main()

{

int n,a[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

c=grade(n,a);

if(c==1)

printf("first grade");

else if(c==2)

printf("second grade");

else

printf("third grade");

return 0;

}

int grade(int n,int a[])

{

int i,sum=0;

float avg;

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

{

sum=sum+a[i];

}

avg=sum/n;

if(avg>80)

{

return 1;

}

else if(avg>60)

{

return 2;

}

else

return 3;

}

**30. generateCode**

Read the question carefully and follow the input and output format.  
  
In a game show everybody got one coupon with some code. They need to generate a code with only even numbers in that coupon. Find the answer.  
  
**Input and Output Format :**  
Input consists of an integer. Output consist of an integer, which is the generated code.  
  
1) Print "Number too small" when the given input number is a negative number.  
2) Print "Number too large" when the given input number is greater than 32767.  
3) Print 0 If the coupon does not contain any even numbers.  
  
Include a function named generateCode(int coupon) whose return type is an integer, which is the generated code.  
  
**Sample Input 1:**  
4352  
  
**Sample Output 1:**  
42  
  
**Sample Input 2:**  
1357  
  
**Sample Output 2:**  
0  
  
**Sample Input 3:**  
-1357  
  
**Sample Output 3:**  
Number too small

#include<stdio.h>

int even(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",even(n));

return 0;

}

int even(int n)

{

int d,sum=0,d1,rev=0;

while(n>0)

{

d=n%10;

if(d%2==0)

{

sum=sum\*10+d;

}

n=n/10;

}

while(sum>0)

{

d1=sum%10;

rev=rev\*10+d1;

sum=sum/10;

}

return rev;

}

**31. maxScoreCount**

Read the question carefully and follow the input and output format.  
  
Student1 and Student2 are of same class and have recieved their scores for different subjects. Given each subject scores of the Student1 & Student2, Find out in how many subjects student1 has scored more marks than student2.  
  
**Input and Output Format :**  
First line of input corresponds to the n, the number of subjects. The next n lines correspond to the scores of Student 1 and the next n lines correspond to the scores of student 2. Output is the number of subjects student1 scored more marks than student2.  
  
Print "Invalid size" when size of the array is a negative number and terminate the program  
Print "Invalid score" when there is any negative score and terminate the program  
  
Include a function named maxScoreCount(int size,int student1[],int student2[]) that returns an integer, the number of subjects student1 scored more marks than student2  
  
**Sample Input 1:**  
7  
45  
23  
67  
34  
88  
13  
67  
33  
56  
89  
44  
67  
89  
55  
  
**Sample Output 1:**  
3  
  
**Sample Input 2:**  
-3  
**Sample Output 2:**  
Invalid size  
  
**Sample Input 3:**  
2  
30  
-60  
**Sample Output 3:**  
Invalid score

#include<stdio.h>

int marks(int n,int a[],int b[]);

int main()

{

int n,a[20],b[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

scanf("%d",&b[i]);

}

printf("%d",marks(n,a,b));

return 0;

}

int marks(int n,int a[],int b[])

{

int i;

int count=0;

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

{

if(a[i]>b[i])

{

count++;

}

}

return count;

}

**32. Sum of squares of prime numbers**

Given an integer n, write a program to find the sum of squares of prime numbers upto and including n.

Include a function named **sumSquarePrime**that accepts an integer argument and returns an integer that corresponds to result. The function returns -1 if the input is less than zero or if the number is greater than 32767.

If the function returns -1, print “Invalid Input”.

Please note that 1 is neither prime nor composite.

**Input and Output Format:**

The input consists of an integer.

The output consists of an integer that corresponds to the sum of the squares of prime numbers.

**Sample Input 1:**

10

**Sample Ouput 1:**

87

**Sample Input 2:**

-8

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int sqrt(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",sqrt(n));

return 0;

}

int sqrt(int n)

{

int i,j,sum=0;

for(i=2;i<=n;i++)

{

for(j=2;j<=i-1;j++)

{

if(i%j==0)

{

break;

}

}

if(i==j)

{

sum=sum+i\*i;

}

}

return sum;

}

**33. clearedStage1**

Read the question carefully and follow the input and output format.  
  
Given an integer array. The first index represents the Student id, Second index represents C-programming marks and the third index Represents SQL marks. Write a program to find the Ids of students who have cleared both C-programming and SQL.  
  
Note :(1) The Pass Marks is >=70  
  
**Input and Output Format :**  
  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer array.  
  
1) Print "Invalid array size" when size of the array is negative and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named clearedStage1(int array[], int size) whose return type is void.  
The output array is stored in a global variable named cleared.  
  
**Sample Input 1:**  
9  
1  
25  
75  
3  
75  
80  
2  
75  
75  
  
**Sample Output 1:**  
3  
2  
  
**Sample Input 2:**  
6  
4  
25  
-78  
  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

void pass(int n,int a[]);

int op[20];

int k;

int main()

{

int n,a[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

pass(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void pass(int n,int a[])

{

int i;

for(i=0;i<n;i=i+3)

{

if(a[i+1] >=70 && a[i+2]>=70)

{

k=k+0;

op[k]=a[i];

k++;

}

}

}

**34. Repeated Element**

Write a program to find the maximum repeated element in a given input array.

Include a function named **maxRepeatedElement** that accepts 2 arguments and returns an int. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The function returns an intthat corresponds to the maximum repeated element.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer that corresponds to the maximum repeated element.

Assume that the maximum number of elements in the array is 20 and that there will always be a unique maximum repeated element.

**Sample Input 1:**

8

2

1

3

4

6

8

10

8

**Sample Output 1:**

8

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int rep(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",rep(n,a));

return 0;

}

int rep(int n,int a[])

{

int i,j,count,op[20],k=0,max1,max;

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

{

count=0;

for(j=i+1;j<n;j++)

{

if(a[i]==a[j])

{

count++;

}

}

op[k]=a[i];

op[k+1]=count;

k=k+2;

}

max=0;

max1=0;

for(i=0;i<k;i=i+2)

{

if(op[i+1]>max)

{

max=op[i+1];

max1=op[i];

}

}

return max1;

}

**46. registerAccountNumbers**

Read the question carefully and follow the input and output format.  
  
Given an array in which the elements are in xxxyy format, where first xxx digits represent the Branch code and the yyrepresents the account  
id. Find out the No of accounts in the given branch code  
  
Input and Output Format :  
The first input n corresponds to the size of the array, the next n lines correspond to the elements of the array and the last line of the input corresponds to the branch code.  
Output corresponds to the number of accounts in the given branch code  
If the given branch code is not available, print 0.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program  
2) Print "Invalid account Number" when there is any negative number available in the input array and terminate the program  
3) Print "Invalid branch code" when branch code is negative number and terminate the program  
  
Include a function named registerAccountNumbers (int size, int account\_numbers[], int branch\_code) that returns the no of accounts  
  
**Sample Input 1 :**  
6  
12345  
12370  
12324  
13355  
13333  
14575  
123  
  
**Sample Output 1 :**  
3  
  
**Sample Input 2 :**  
-6  
  
**Sample Output 2:**  
Invalid array size

#include<stdio.h>

int reg(int n,int a[],int d);

int main()

{

int n,a[20],d,i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

scanf("%d",&d);

printf("%d",reg(n,a,d));

return 0;

}

int reg(int n,int a[],int d)

{

int i,count=0;

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

{

if(a[i]/100==d)

{

count++;

}

}

return count;

}

**47. Sort and Insert**

Write a program to insert the given number in the appropriate place after sorting the input array in ascending order and store the sorted array in the output array.

Include a function named **sortAndInsert** that accepts 3 arguments and its return type is void. The first argument is the input array, the second argument is an int that corresponds to the size of the array and the third argument is the array element to be inserted. The output array is stored in a global variable named output1.

If the size of the array is negative or if any of the elements in the array are negative , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+2 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array. The last integer corresponds to the element to be inserted.

Output consists of an integer array.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

1

6

3

5

8

10

4

9

2

**Sample Output 1:**

1

2

3

4

5

6

8

9

10

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**48. secondLargest**

Read the question carefully and follow the input and output format.  
  
Write a function to find second largest number in the given input integer array.  
  
Assume that all elements in the input array are unique.  
  
**Input and Output Format :**  
  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements.  
Output consist of an integer, which is the second largest.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named secondLargest(int array[], int size) whose return type is an integer, the second largest.  
  
**Sample Input 1:**  
5  
3  
342  
53  
2  
12  
  
**Sample Output 1:**  
53  
  
**Sample Input 2:**  
5  
3  
342  
53  
-2  
  
  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int large(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",large(n,a));

return 0;

}

int large(int n,int a[])

{

int i,t,j,max;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]<a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

max=a[1];

return max;

}

**49. 3/5 Number**

Write a program to find whether the given number is a 3/5 Number.

A number is a 3/5 Number if the product of the digits in the number is divisible by 3 or 5.

Include a function named **divisibleByThreeFive** that accepts an integer argument and returns an integer. The function returns

1. 1 if it is a 3/5 Number

2. 0 if it is not a 3/5 Number

3. -1 if it is a negative number

**Input and Output Format:**

Input consists of a single integer.

Output consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

251

**Sample Output 1:**

yes

**Sample Input 2:**

241

**Sample Output 2:**

no

**Sample Input 3:**

-9

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int five(int n);

int main()

{

int n,c;

scanf("%d",&n);

c=five(n);

if(c==1)

printf("yes");

if(c==2)

printf("no");

return 0;

}

int five(int n)

{

int pro=1,d;

while(n>0)

{

d=n%10;

pro=pro\*d;

n=n/10;

}

if(pro%3==0 || pro%5==0)

{

return 1;

}

else

return 2;

}

**50. findMileage**

Read the question carefully and follow the input and output format.  
  
Given the cubic capacity(CC) of a bike. Write a function to return the mileage/liter for the given Cubic Capacity(CC). The mileage will be calculated as follows:  
  
if CC is between 100 and 125, mileage is 75  
if CC is between 126 and 135, mileage is 70  
if CC is between 136 and 150, mileage is 60  
if CC is between 151 and 200, mileage is 50  
if CC is between 201 and 220, mileage is 35  
  
First line of input consists of an integer that corresponds to CC of a bike. Output consist of an integer, which is the mileage.  
  
Print "Number too large" when the given input CC is greater than 220.  
Print "Number too small" when the given input CC is less than 100.  
  
Include a function named findMileage(int cc) whose return type is an integer, which is the mileage.  
  
**Sample Input 1:**  
1  
  
**Sample Output 1:**  
Number too small  
  
**Sample Input 2:**  
160  
  
**Sample Output 2:**  
50

#include<stdio.h>

int mile(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",mile(n));

return 0;

}

int mile(int n)

{

int mil;

if(n>=120 && n<=125)

{

mil=75;

}

if(n>=126&& n<=135)

{

mil=70;

}

if(n>=136 && n<=150)

{

mil=60;

}

if(n>=151 && n<=200)

{

mil=50;

}

if(n>=201 && n<=220)

{

mil=35;

}

return mil;

}

**51. Decimal Conversion**

Write a program to convert a given input binary number to decimal.

Include a function named **convertToDecimal** that accepts an integer argument and returns an integer that corresponds to the decimal representation of the input number. If the input value is not a binary value or if the input is negative or if the input is greater than 11111, the function returns -1.

If the function returns -1, print Invalid Input.

**Input and Output Format:**

Input consists of a single integer that corresponds to the binary representation of a number.

Output consists of a single integer that corresponds to the decimal equivalent of the given number.

Refer sample output for formatting specifications.

**Sample Input 1:**

1100

**Sample Output 1:**

12

**Sample Input 2:**

101010

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

1201

**Sample Output 3:**

Invalid Input

#include<stdio.h>

#include<math.h>

int decimal(int n);

int main()

{

int n,c;

scanf("%d",&n);

c=decimal(n);

if(c==-1)

{

printf("invalid");

return 0;

}

else

printf("%d",c);

return 0;

}

int decimal(int n)

{

int i=0,d,c1=0,sum=0;

if(n>11111)

{

return -1;

}

while(n>0)

{

d=n%10;

if(d>1)

{

return -1;

break;

}

else

{

c1=d\*(pow(2,i));

sum=sum+c1;

n=n/10;

i++;

}

}

return sum;

}

**52. Sum of positive numbers in Array**

Write a program to find the sum of positive numbers in an array.

Include a function named**addPositives** that accepts 2 arguments and returns an int. The first argument is the input array and the second argument is an int that corresponds to the size of the array. The function returns the sum of the positive numbers in the array.

If the size of the array is negative, print “Invalid Input” and terminate the program..

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Assume that the maximum size of the array is 20.

**Sample Input 1:**

5

3

5

-2

6

-6

**Sample Output 1:**

14

**Sample Input 2:**

**-5**

**Sample Output 2:**

Invalid Input

**61. passCount**

Read the question carefully and follow the input and output format.  
  
Given a input array, First index Represents RollNo second index represents Mark and so on. Write a program to find the number of students who had cleared the exam.  
  
Note : If marks >=70 then He /she Cleared the exam. Array size is always even.  
  
**Input and Output Format :**  
  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer,  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named passCount(int array[], int size) whose return type is an integer , the count.  
  
**Sample Input 1:**  
8  
1  
70  
2  
55  
3  
75  
4  
80  
  
**Sample Output 1:**  
3  
  
**Sample Input 2:**  
5  
6  
2  
8  
-2  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int count(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",count(n,a));

return 0;

}

int count(int n,int a[])

{

int i,c=0;

for(i=0;i<n;i=i+2)

{

if(a[i+1]>=70)

{

c++;

}

}

return c;

}

**62. dailyAllowance**

Read the question carefully and follow the input and output format.  
  
A Sales person daily allowances calculated as follows .  
Item Money (rupees)  
Shirt 15  
Saree 10  
other items 5  
  
Given an input array in which the first index represents no.of shirts sold, second index represents the no of sareessold and the third index represents the other items sold for a particular day, Calculate the total allowances.  
  
Inlcude a function named dailyAllowance(int items[], int size) that returns an integer, which is the total allowances.  
  
Business Rules:  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program  
2) Print "Invalid item count" when there is any negative numbers available in the input array and terminate the program  
3) Print "Array size greater than 3" when size of the array is greater than 3 and terminate the program.  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of the total allowance.  
  
**Sample Input 1**:  
3  
10  
5  
10  
**Sample Output 1:**  
250  
  
**Sample Input 2:**  
4  
  
**Sample Output 2:**  
Array size greater than 3

#include<stdio.h>

int shirt(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",shirt(n,a));

return 0;

}

int shirt(int n,int a[])

{

int x,y,z,sum;

x=a[0]\*15;

y=a[1]\*10;

z=a[2]\*5;

sum=x+y+z;

return sum;

}

**63. primeFactorialSum**

Read the question carefully and follow the input and output format.  
  
In a given input number , find out the sum of factorial of digits that are prime.  
  
**Input and Output Format :**  
Input consists of an integer. Output consists of the factorial sum.  
1) Print "Number too large" when the given input number is greater than 32767  
2) Print "Number too small" when the given input number is a negative number.  
  
Include a function named primeFactorialSum(int number) whose return type is an integer.  
  
**Sample Input 1:**  
123  
**Sample Output 1:**  
8  
  
Hint : 2! + 3! = (8)  
  
**Sample Input 2:**  
32768  
**Sample Output 2:**  
Number too large

#include<stdio.h>

int prime(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",prime(n));

return 0;

}

int prime(int n)

{

int d,i,j,fact=1,sum=0,flag;

while(n>0)

{

d=n%10;

fact=1;

flag=0;

if(d==1)

{

flag=1;

break;

}

for(i=2;i<d;i++)

{

if(d%i==0)

{

flag=1;

break;

}

}

if(flag==0)

{

for(j=1;j<=d;j++)

{

fact=fact\*j;

}

sum=sum+fact;

}

n=n/10;

}

return sum;

}

**65. avgOddKeyValues**

Read the question carefully and follow the input and output format.  
  
  
Given an input array, First index represents key and second index represents the value and so on... Write code to find out the average of all values whose keys are odd numbers.  
  
**Input and Output Format :**  
First line of input consists of n, the next n lines correspond to the elements of the array. Output consist of the an integer.  
  
Print "Invalid array size" when size of the array is a negative number and terminate the program.  
Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named avgOddKeyValues(int numbers[], int size) whose return type is an integer.  
  
**Sample Input 1:**  
8  
1  
3  
2  
4  
3  
16  
4  
25  
**Sample Output 1:**  
9  
  
**Sample Input 2:**  
-3  
**Sample Output 2:**  
Invalid array size

#include<stdio.h>

int sum(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",sum(n,a));

return 0;

}

int sum(int n,int a[])

{

int s=0,i;

float count=0;

float avg;

for(i=0;i<n;i=i+2)

{

if(a[i]==1 || a[i]%2!=0)

{

s=s+a[i+1];

count++;

}

}

avg=s/count;

return avg;

}

**70. Product of Prime Digits**

Write a program to find the product of the prime digits in the given input number.

Include a function named **productPrimeDigits** that accepts an integer argument and returns an integer that corresponds to the product of the prime digits in the integer.

The function returns -1 if the input number is negative or greater than 32767.

If the function returns -1, print Invalid Input.

Please note that 1 and 0 are neiher prime nor composite.

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

324

**Sample Output 1:**

6

**Sample Input 2:**

-67

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int prime(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",prime(n));

return 0;

}

int prime(int n)

{

int d,flag,pro=1,i;

while(n>0)

{

d=n%10;

flag=0;

for(i=2;i<d;i++)

{

if(d%i==0)

{

flag=1;

pro=0;

break;

}

}

if(flag==0)

{

pro=pro\*d;

}

n=n/10;

}

return pro;

}

**74. Array Multiplication in Reverse**

A company wanted to know the reward points of the employee so that at the end of every month they will credit some amount along with their salary. Each employee has 2 separate lists, in first list records will be sorted in employee’s employee number in ascending order. Second list records will be sorted in employee’s employee number in descending order. Hence the management has decided to multiply both the reward points and credit the amount based on the points. Here they followed the formula for multiplying the first entry value in the first list with the last entry value in the second list and second entry from the first list with the second last record from the second list. Repeat the same for all the entries in the lists.

Include a function named **arrayProduct** that accepts 3 arguments and ints return type is void. The first argument is the input array 1, the second argument is the input array 2 and the third argument is an int that corresponds to the size of the array. The output array is stored in a global variable named output1.

If the size of the array is negative or if any element in any of the array is negative , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of utmost 2n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the first array. The last 'n' integers correspond to the elements in the second array. If any of the inputs are invalid, then terminate the program.

Output consists of n integers that correspond to the elements in the result array.

Assume that the maximum size of the array is 20.

**Sample Input 1 :**

5

23

2

5

32

76

2

2

21

42

4

**Sample Output 1 :**

92

84

105

64

152

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-5

**Sample Output 3:**

Invalid Input

#include<stdio.h>

void array(int n,int a[],int b[]);

int op[20];

int k;

int main()

{

int n,a[20],b[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

scanf("%d",&b[i]);

}

array(n,a,b);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void array(int n,int a[],int b[])

{

int i;

k=0;

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

{

op[k]=a[i]\*b[n-i-1];

k++;

}

}

**75. generateNewNumber**

Read the question carefully and follow the input and output format.  
  
Write a program to generate new number from the given input based on following conditions.  
  
(i) Even digit should be replaced by next Even digit.  
(ii) Odd digit should be replaced with next Odd digit  
  
**Input and Output Format :**  
Input consists of an integer. Output is also an integer.  
  
1) Print "Number too small" when any of given input numbers is a negative number.  
2) Print "Number too large" when any of given input numbers is greater than 32767.  
  
Include a function named generateNewNumber(int number) whose return type is an integer, which is the replaced number.  
  
**Sample Input 1:**  
123  
  
**Sample Output 1:**  
345  
  
**Sample Input 2:**  
32768  
  
**Sample Output 2:**  
Number too large

#include<stdio.h>

int new(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",new(n));

return 0;

}

int new(int n)

{

int d,sum=0,d1,rev=0;

while(n>0)

{

d=n%10;

sum=2+d+(sum\*10);

n=n/10;

}

while(sum>0)

{

d1=sum%10;

rev=rev\*10+d1;

sum=sum/10;

}

return rev;

}

**76. newArraySum**

Read the question carefully and follow the input and output format.  
  
Given an input array which contains age of some employees, write a program t fund the sum of ages of employees greater than 18.  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer, which is the sum.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named newArraySum(int age[],int size) whose return type is an integer, which is the sum.  
  
**Sample Input 1:**  
5  
21  
22  
17  
10  
25  
**Sample Output 1:**  
68  
  
**Sample Input 2:**  
6  
50  
-36  
  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int age(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",age(n,a));

return 0;

}

int age(int n,int a[])

{

int i,sum=0;

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

{

if(a[i]>=18)

{

sum=sum+a[i];

}

}

return sum;

}

**81. Descending Order Sort**

Write a program to sort the given array in descending order.

Include a function named **sortArray** that accepts 2 arguments and its return type is void. The first argument is the input array and the second argument is an int that corresponds to the size of the array .

If the size of the array is negative or if any of the elements in the array are negative , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer array.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

1

6

3

5

8

10

4

9

**Sample Output 1:**

10

9

8

6

5

4

3

1

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

#include<stdio.h>

void desc(int n,int a[]);

int op[20];

int k;

int main()

{

int n,a[20],c,i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

desc(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void desc(int n,int a[])

{

int i,t,j;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]<a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

k=0;

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

{

op[k]=a[i];

k++;

}

}

**82. adjecentDifference**

Read the question carefully and follow the input and output format.  
  
Given an input Integer array, find the difference in the adjacent elements and print the largest difference  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of largest adjacent difference.  
  
Print "Invalid array size" when size of the array is a negative number and terminate the program  
Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named adjecentDifference(int numbers[], int size) whose return type is an integer  
  
**Sample Input 1:**  
7  
2  
4  
5  
1  
9  
3  
8  
**Sample Output 1:**  
8  
  
Hint: The AdjecentElement Diff are :2,1,4,8,6,5 and Maximum diff is 8 which is obtained by 2-4 =2, 4-5=1,5- 1 =4, 1-9=8,9-3=6,3-8 =5  
  
**Sample Input 2:**  
5  
1  
7  
3  
-8  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int diff(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",diff(n,a));

return 0;

}

int diff(int n,int a[])

{

int i,max,dif;

max=0;

for(i=0;i<n-1;i++)

{

if(a[i]>a[i+1])

{

dif=a[i]-a[i+1];

}

if(a[i+1]>a[i])

{

dif=a[i+1]-a[i];

}

if(dif>max)

{

max=dif;

}

}

return max;

}

**83. findGrade**

Read the question carefully and follow the input and output format.  
  
In a school examination the result of students are published in the form of an array where first index is the student id and the second index is the total marks in mathematics third index is student id and fourth index is total marks in mathematics and so on... Write a method that assigns the student id as the key and the grade in mathematics as the value to the output array based on the following conditions:  
  
If(Marks >=90 ) : 1  
If(Marks >=80 and <90 ) : 2  
If(Marks >=70 and <80) : 3  
If(Marks <70 ) : 0  
  
Hint: Array size is always is even.  
  
**Input and Output Format :**  
  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer array.  
1) Print "Invalid array size" when size of the array is negative and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named findGrade(int array[], int size) whose return type is void.  
The output array is stored in a global variable named grade.  
  
**Sample Input 1:**  
10  
1  
65  
2  
74  
3  
86  
4  
95  
6  
69  
**Sample Output 1:**  
1  
0  
2  
3  
3  
2  
4  
1  
6  
0  
  
**Sample Input 2:**  
4  
-3  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

void grade(int n,int a[]);

int op[20];

int k;

int main()

{

int n,a[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

grade(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void grade(int n,int a[])

{

int i;

k=0;

for(i=0;i<n-1;i=i+2)

{

if(a[i+1]>=70&&a[i+1]<=80)

{

op[k]=a[i];

op[k+1]=3;

k=k+2;

}

if(a[i+1]>=81&&a[i+1]<=90)

{

op[k]=a[i];

op[k+1]=2;

k=k+2;

}

if(a[i+1]>90)

{

op[k]=a[i];

op[k+1]=1;

k=k+2;

}

if(a[i+1]<70)

{

op[k]=a[i];

op[k+1]=0;

k=k+2;

}

}

}

**59. Strong Number**

Write a program to find whether the given input number is a Strong Number

Strong Number : (In a number sum of Factorial of individual digits equals to the same number).

Include a function named **checkStrong** that accepts an integer and returns an integer. The function returns

1. 1 if the input is a Strong Number

2. 0 if the input is not a Strong Number

3. -1 if the input is a negative number

Print Invalid Input if the function returns -1.

**Input and Output Format:**

Input consists of a single integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

145

**Sample Output 1:**

yes

**Sample Input 2:**

141

**Sample Output 2:**

no

**Sample Input 3:**

**-**2345

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int strong(int n);

int main()

{

int n,c;

scanf("%d",&n);

c=strong(n);

if(c==1)

printf("yes");

else

printf("no");

return 0;

}

int strong(int n)

{

int sum=0,fact,d,i;

int a;

a=n;

while(n>0)

{

fact=1;

d=n%10;

for(i=1;i<=d;i++)

{

fact=fact\*i;

}

sum=sum+fact;

n=n/10;

}

if(sum==a)

return 1;

else

return -1;

}

**61. Sum of Prime Numbers**

Write a program to find the sum of the prime numbers present in the given input array.

Include a function named**sumPrime**that accepts 2 arguments and returns an int. The first argument is a pointer to the input array and the second argument is an int that corresponds to the size of the array. The function returns the sum of the prime numbers in the input array.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

Please note that 1 is neither prime nor composite.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

5

2

4

8

9

11

**Sample Output 1:**

13

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int prime(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",prime(n,a));

return 0;

}

int prime(int n,int a[])

{

int i,j,flag,sum=0;

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

{

flag=0;

if(a[i]==1)

{

flag++;

}

for(j=2;j<a[i];j++)

{

if(a[i]%j==0)

{

flag=1;

break;

}

}

if(flag==0)

{

sum=sum+a[i];

}

}

return sum;

}

**64. convertToBinary**

Read the question carefully and follow the input and output format.  
  
Kate is a military officer. He needs to send top-secret code to the other soldiers in a different place. He need to encrypt it. We need to write a function to convert the given decimal number to the corresponding binary number.  
  
Input consist of single integer. Output consists of binary number.  
  
1) Print "Number too small" when input is negative.  
2) Print "Number too large" when input value is greater than 100.  
  
Include a function named convertToBinary(intnum) whose return type is void.  
  
**Sample Input 1:**  
12  
  
**Sample Output 1:**  
1100  
  
**Sample Input 2:**  
101  
  
**Sample Output 2:**  
Number too large

#include<stdio.h>

void binary(int n);

int op[20];

int k;

int main()

{

int n,c,i;

scanf("%d",&n);

binary(n);

c=k;

for(i=c-1;i>=0;i--)

{

printf("%d",op[i]);

}

return 0;

}

void binary(int n)

{

int d;

k=0;

while(n>0)

{

d=n%2;

op[k]=d;

k++;

n=n/2;

}

}

**65. Interchange Array**

Write a program to interchange the first element in the array with the last element in the array. Repeat the process till the middle of the array.

Include a function named **interchangeArray** that accepts 2 arguments and its return type is void. The first argument is the input array and the second argument is an int that corresponds to the size of the array.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of the interchanged array.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

4

2

1

3

4

**Sample Output 1:**

4

3

1

2

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

void rev(int n,int a[]);

int op[20];

int k;

int main()

{

int n,a[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

rev(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void rev(int n,int a[])

{

int i;

k=0;

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

{

op[k]=a[n-i-1];

k++;

}

}

**69. newNumber**

Read the question carefully and follow the input and output format.  
  
Write a program to find the difference between consecutive digits in the given input integer and display it.  
  
**Input and Output Format:**  
Input consists of an integer and output the difference between the consecutive digits.  
  
Print "Number too small" if the number is less than 0  
Print "Number too large" if the number is greater than 32767  
  
Include a function named newNumber(int number) that returns a integer  
  
**Sample Input 1:**  
1325  
**Sample Output 1:**  
213  
  
**Sample Input 2:**  
-13  
**Sample Output 2:**  
Number too small

#include<stdio.h>

int new(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",new(n));

return 0;

}

int new(int n)

{

int d,d1,d2,rev=0,dif,dif1,sum=0;

while(n>10)

{

d=n%10;

n=n/10;

d1=n%10;

if(d>d1)

{

dif=d-d1;

sum=sum\*10+dif;

}

else

{

dif1=d1-d;

sum=sum\*10+dif1;

}

}

while(sum>0)

{

d2=sum%10;

rev=rev\*10+d2;

sum=sum/10;

}

return rev;

}

**73. sumThreeLargest**

Read the question carefully and follow the input and output format.  
  
Write a program to find the sum of first ,second and third largest element in the given array.  
  
**Input and Output Format:**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements . Output consists of an Integer, the sum.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named sumThreeLargest(int array[], int n) whose return type is integer  
  
**Sample Input 1:**  
8  
1  
2  
2  
3  
4  
5  
5  
4  
  
**Sample Output 1:**  
12  
  
**Sample Input 2:**  
4  
1  
2  
-3  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int large(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",large(n,a));

return 0;

}

int large(int n,int a[])

{

int i,j,b[20],sum,k,t;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]==a[j])

{

a[j]=-1;

}

}

}

j=0;

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

{

if(a[i]>0)

{

b[j]=a[i];

j++;

}

}

for(i=0;i<j;i++)

{

for(k=i+1;k<j;k++)

{

if(b[i]<b[k])

{

t=b[i];

b[i]=b[k];

b[k]=t;

}

}

}

sum=b[0]+b[1]+b[2];

return sum;

}

**75. Sort and Delete**

Write a program to delete the given number in the input array and then to sort the array.

Include a function named **sortAndDelete** that accepts 3 arguments and its return type is void. The first argument is the input array and the second argument is an int that corresponds to the size of the array and the third argument is the array element to be deleted. The number of elements in the modified array is stored in the global variable named output1.

If the size of the array is negative or if any of the elements in the array are negative , print “Invalid Input” and terminate the program.

Please note that the elements in the array may not be unique.

**Input and Output Format:**

Input consists of n+2 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array. The last integer corresponds to the element to be deleted.

Output consists of an integer array.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

1

6

3

5

8

10

4

8

8

**Sample Output 1:**

1

3

4

5

6

10

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

#include<stdio.h>

void sort(int n,int a[],int d);

int op[20];

int k;

int main()

{

int n,a[20],i,d,c,j,t;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

scanf("%d",&d);

sort(n,a,d);

c=k;

for(i=0;i<k;i++)

{

for(j=i+1;j<k;j++)

{

if(op[i]>op[j])

{

t=op[i];

op[i]=op[j];

op[j]=t;

}

}

}

for(i=0;i<k;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void sort(int n,int a[],int d)

{

int i;

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

{

if(a[i]==d)

{

a[i]=-1;

}

}

k=0;

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

{

if(a[i]>0)

{

op[k]=a[i];

k++;

}

}

}

**76. commonElementsSum**

Read the question carefully and follow the input and output format.  
  
Given 2 integer arrays , write a program to find the sum of common elements in both the arrays.  
  
If there are no common elements print 0.  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the first array elements and the next n lines correspond to the second array elements. Output consist of an integer, which is the sum  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named commonElementsSum(int elements1[],int elements2[],int size) whose return type is an integer, the sum.  
  
**Sample Input 1:**  
4  
1  
2  
3  
4  
2  
3  
6  
7  
  
**Sample Output 1:**  
5  
  
**Sample Input 2:**  
3  
8  
6  
-7  
  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int sum(int n,int a[],int b[]);

int main()

{

int n,a[20],b[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

scanf("%d",&b[i]);

}

printf("%d",sum(n,a,b));

return 0;

}

int sum(int n,int a[],int b[])

{

int i,j,sum=0;

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

{

for(j=0;j<n;j++)

{

if(a[i]==b[j])

{

sum=sum+a[i];

}

}

}

return sum;

}

**77. primeIndexSum**

Read the question carefully and follow the input and output format.  
  
Given an Integer array. Find the average of the numbers located on the Prime Indexes of the Array. Consider 0 index as 1 and 1 index is 2 and so on……  
  
Hint :Consider 1 is not a prime number  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements . Output consists of an Integer, the prime index sum.  
  
1) Print "Invalid array size" when size of the array is a negative number.  
2) Print "Invalid input" when there is any negative numbers available in the input array.  
  
Include a function named primeIndexSum(int array[], int size) whose return type is an integer,which is the sum.  
  
**Sample Input 1:**  
7  
2  
4  
5  
1  
9  
3  
8  
  
**Sample Output 1:**  
6  
  
**Sample Input 2:**  
-7  
  
**Sample Output 2:**  
Invalid array size

 #include<stdio.h>

int prime(int a[],int n);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",prime(a,n));

return 0;

}

int prime(int a[],int n)

{

int i,j,flag,sum=0,avg,count=0;

for(i=1;i<=n;i++)

{

flag=0;

if(i==1)

{

flag=1;

}

for(j=2;j<i;j++)

{

if(i%j==0)

{

flag=1;

break;

}

}

if(flag==0)

{

sum=sum+a[i-1];

count=count+1;

}

}

avg=sum/count;

return avg;

}

**79. firstMiddleSame**

Read the question carefully and follow the input and output format.  
  
Write a program to check if the first and middle element in an array is the same, if so display “100” in output or else display the first element of the array.  
  
Note: Array size is always odd.  
  
**Input and Output Format :**  
The first line of the input consists of an integer, n that corresponds to the number of elements in the array.  
The next 'n' lines correspond to the elements in the array.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
3) Print "Size must be odd" when the size of the array is even.  
  
Include a function named firstMiddleSame(int array[], int size) whose return type is an integer..  
  
**Sample Input 1:**  
7  
8  
4  
5  
8  
3  
2  
1  
  
**Sample Output 1:**  
100  
  
**Sample Input 2:**  
10  
4  
5  
6  
-8  
  
**Sample Output 2:**  
Invalid input  
  
**Sample Input 3:**  
3  
4  
8  
6  
  
**Sample Output 3:**  
4

#include<stdio.h>

int mid(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",mid(n,a));

return 0;

}

int mid(int n,int a[])

{

int i;

if(a[0]==a[n/2])

return 100;

else

return -1;

}

**81. aboveAverageMarks**

Read the question carefully and follow the input and output format.  
  
Given an input array that represents the marks of students, find out the marks which are greater than or equal to average mark of all students.  
  
**Input and Output Format:**  
First line of input consists of n, the number of elements in the input array.  
Next n lines correspond to the array elements. Output consist of an integer array.  
  
1) Print "Invalid array size" when size of the array is negative and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named aboveAverageMarks(int array[], int size) whose return type is void.  
The output array is stored in a global variable named above\_average.  
  
**Sample Input 1:**  
5  
10  
20  
30  
40  
50  
**Sample Output 1:**  
30  
40  
50  
  
**Sample Input 2:**  
4  
-3  
**Sample Output 2:**  
Invalid Input

#include<stdio.h>

void avg(int n,int a[]);

int op[20];

int k;

int main()

{

int n,a[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

avg(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void avg(int n,int a[])

{

int i,sum=0,avg;

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

{

sum=sum+a[i];

}

avg=sum/n;

k=0;

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

{

if(a[i]>=avg)

{

op[k]=a[i];

k++;

}

}

}

**82. avgOddKeyValues**

Read the question carefully and follow the input and output format.  
  
  
Given an input array, First index represents key and second index represents the value and so on... Write code to find out the average of all values whose keys are odd numbers.  
  
**Input and Output Format :**  
First line of input consists of n, the next n lines correspond to the elements of the array. Output consist of the an integer.  
  
Print "Invalid array size" when size of the array is a negative number and terminate the program.  
Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named avgOddKeyValues(int numbers[], int size) whose return type is an integer.  
  
**Sample Input 1:**  
8  
1  
3  
2  
4  
3  
16  
4  
25  
**Sample Output 1:**  
9  
  
**Sample Input 2:**  
-3  
**Sample Output 2:**  
Invalid array size

#include<stdio.h>

int avg(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",avg(n,a));

return 0;

}

int avg(int n,int a[])

{

int i,sum=0,aveg,count=0;

for(i=0;i<n-1;i=i+2)

{

if(a[i]==1 || a[i]%2!=0)

{

sum=sum+a[i+1];

count++;

}

}

aveg=sum/count;

return aveg;

}

**88. sortCommonElements**

Read the question carefully and follow the input and output format.  
  
Find out the common elements in the given input arrays and sort the common elements in ascending order.  
  
**Input and Output Format :**  
First line corresponds to n, the size of the array. The next n lines correspond to elements in the first array. The next n lines correspond to the elements in the second array. Output corresponds to the common elements sorted in ascending order.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program  
  
Include a function named sortCommonElements(int set1[],int set2[],int size) whose return type is void.  
The output array is stored in a global variable named common.  
  
**Sample Input 1:**  
5  
1  
3  
2  
5  
7  
4  
5  
6  
7  
2  
  
**Sample Output 1:**  
2  
5  
7  
  
**Sample Input 2:**  
5  
5  
8  
-9  
  
**Sample Output 2:**  
Invalid input  
  
**Sample Input 3:**  
-4  
  
**Sample Output 3:**  
Invalid array size

#include<stdio.h>

void comm(int n,int a[],int b[]);

int op[20];

int k;

int main()

{

int n,a[20],b[20],i,j,c,t;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

scanf("%d",&b[i]);

}

comm(n,a,b);

c=k;

for(i=0;i<c;i++)

{

for(j=i+1;j<c;j++)

{

if(op[i]>op[j])

{

t=op[i];

op[i]=op[j];

op[j]=t;

}

}

}

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void comm(int n,int a[],int b[])

{

int i,j;

k=0;

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

{

for(j=0;j<n;j++)

{

if(a[i]==b[j])

{

op[k]=a[i];

k++;

}

}

}

}

**90. Largest Array**

Write a program which takes two arrays of the same size as a input and compares the first element of first array with the first element of second array and stores the largest of these into the first element of the output array. Repeat the process till the last element of the first array is checked with the last element of the second array.

Include a function named **largestArray** that accepts 3 arguments and its return type is void. The first argument is input array 1, the second argument is input array 2 and the third argument is an int that corresponds to the size of the array. The output array is stored in a global variable named output1.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of the largest array.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

4

2

1

3

4

1

9

2

8

**Sample Output 1:**

2

9

3

8

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

**Sample Input 3:**

5

23

2

-200

**Sample Output 3:**

Invalid Input

#include<stdio.h>

void large(int n,int a[],int b[]);

int op[20];

int k;

int main()

{

int n,a[20],b[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

scanf("%d",&b[i]);

}

large(n,a,b);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void large(int n,int a[],int b[])

{

int i;

k=0;

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

{

if(a[i]>b[i])

{

op[k]=a[i];

k++;

}

else

{

op[k]=b[i];

k++;

}

}

}

**97. subTwoArrays**

Read the question carefully and follow the input and output format.  
  
Given two input arrays, write a program to find out numbers which is present in the first array and not in the second array.  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the first array elements and the next n lines correspond to the second array elements. Output consist of an integer array.  
  
1) Print "Invalid array size" when size of the array is a negative number.  
2) Print "Invalid input" when there is any negative numbers available in the input array.  
  
  
Include a function named subTwoArrays(int elements1[], int elements2[], int size) whose return type is void.  
The output array is stored in a global variable named no\_common.  
  
**Sample Input 1:**  
5  
1  
2  
3  
4  
5  
3  
5  
7  
9  
10  
  
**Sample Output 1:**  
1  
2  
4  
  
**Sample Input 2:**  
4  
1  
2  
-3  
  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

void comm(int n,int a[],int b[]);

int op[20];

int k;

int main()

{

int n,a[20],b[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

scanf("%d",&b[i]);

}

comm(n,a,b);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void comm(int n,int a[],int b[])

{

int i,j;

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

{

for(j=0;j<n;j++)

{

if(a[i]==b[j])

{

a[i]=-1;

}

}

}

k=0;

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

{

if(a[i]>0)

{

op[k]=a[i];

k++;

}

}

}

**98. Leap Year**

Write a program to find whether the given input year is a Leap Year.

Include a function named **checkLeapYear** that accepts an integer and returns an integer. The function returns

1. 1 if the input is a Leap Year

2. 0 if the input is not a Leap Year

3. -1 if the input is a negative number

Print Invalid Input if the function returns -1.

**Input and Output Format:**

Input consists of a single integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

2000

**Sample Output 1:**

yes

**Sample Input 2:**

1610

**Sample Output 2:**

no

**Sample Input 3:**

**-**2345

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int leap(int n);

int main()

{

int n,c;

scanf("%d",&n);

c=leap(n);

if(c==1)

printf("yes");

else

printf("no");

return 0;

}

int leap(int n)

{

if(n%4==0)

{

return 1;

}

else

return 2;

}

**100. Palindromic Number**

Write a program to find whether the given input number is a palindrome.

Include a function named **checkPalindrome** that accepts an integer and returns an integer. The function returns

1. 1 if the input is a palindrome

2. 0 if the input is not a palindrome

3. -1 if the input is a negative number

Print Invalid Input if the function returns -1.

**Input and Output Format:**

Input consists of a single integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

2002

**Sample Output 1:**

yes

**Sample Input 2:**

167

**Sample Output 2:**

no

**Sample Input 3:**

**-**2345

**Sample Output 3:**

Invalid Input

#include<stdio.h>

int pal(int n);

int main()

{

int c,n;

scanf("%d",&n);

c=pal(n);

if(c==1)

printf("yes");

else

printf("no");

return 0;

}

int pal(int n)

{

int d,a,rev=0;

a=n;

while(n>0)

{

d=n%10;

rev=rev\*10+d;

n=n/10;

}

if(rev==a)

return 1;

else

return 2;

}

**101. changeNumber**

Read the question carefully and follow the input and output format.  
  
Tom needs to generate a new number from the given input with the following conditions.Consider Input is always a 3 digit number.  
  
conditions:  
(i) Middle digit comes first.  
(ii) Last digit should come in middle  
(iii) First digit should come as a last digit  
  
Business rule:  
1. Print "Invalid input" if input is negative number.  
2. Print "Not a 3 digit number" if the given number is not a 3 digit number.  
  
Include a function named changeNumber(int number) that returns an integer.  
  
**Input and Output Format:**  
Input consists of an integer.  
Refer business rules and sample output for output format.  
  
**Sample Input 1:**  
123  
**Sample Output 1:**  
231  
  
  
**Sample Input 2:**  
1234  
**Sample Output 2:**  
Not a 3 digit number

#include<stdio.h>

int new(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",new(n));

return 0;

}

int new(int n)

{

int d,a,c;

d=n%100;

a=n/100;

c=d\*10+a;

return c;

}

**102. Find Index**

Write a program to find the index of a particular number in a given input array.

Include a function named **findIndex** that accepts 3 arguments and returns an int. The first argument is the input array, the second argument is an int that corresponds to the size of the array and the third argument is the element to be searched for. The function returns the corresponding index if the search element is present in the array and returns -1 if the search element is not present in the array.

If the size of the array is negative or if any element in the array is negative, print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+2 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array. The last integer corresponds to the element whose count needs to be found.

Output consists of an integer that corresponds to the index of the search element if it is present.

Else, print 'not found'.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20 and that all elements in the array are unique.

**Sample Input 1:**

8

2

1

3

8

6

12

10

19

8

**Sample Output 1:**

3

**Sample Input 2:**

8

2

1

3

8

6

12

10

19

80

**Sample Output 2:**

not found

**Sample Input 3:**

-5

**Sample Output 3:**

Invalid Input

**Sample Input 4:**

5

23

2

-200

**Sample Output 4:**

Invalid Input

#include<stdio.h>

int comm(int n,int a[],int d);

int main()

{

int n,a[20],d,i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

scanf("%d",&d);

printf("%d",comm(n,a,d));

return 0;

}

int comm(int n,int a[],int d)

{

int i,f;

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

{

if(a[i]==d)

{

f=i;

}

}

return f;

}

**105. highestFeedBack**

Read the question carefully and follow the input and output format.  
  
In a company there are some managers working on two different projects (MetLife and Hardfort). When the feedback was taken their feedback was present in both MetLife Feedback as well as Hardfort Feedback. Write a method to create a consolidated feedback for the managers for MetLife and HardForts. For those working on both the projects the highest feedback is taken. In the 2 given arrays, the First Index represents the Employee id and second one Represents The Feed Back Score and so on....  
  
**Input and Output Format:**  
First line corresponds to n, the size of the array. The next n lines correspond to the elements of the first array. The next n lines correspond to the elements in the second array. Output corresponds to the consolidated feedback score.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program  
  
Include a function named highestFeedBack(intmetlife[],inthardfort[],int size) whose return type is void.  
The output array is stored in a global variable named fedback.  
  
**Sample Input 1:**  
8  
1  
90  
2  
75  
3  
92  
5  
85  
1  
80  
2  
85  
3  
80  
4  
85  
**Sample Output 1:**  
1  
90  
2  
85  
3  
92  
5  
85  
4  
85  
  
**Sample Input 2:**  
5  
5  
8  
9  
1  
-6  
**Sample Output 2:**  
Invalid number  
  
**Sample Input 3:**  
-4  
**Sample Output 3:**  
Invalid array size

#include<stdio.h>

void fdbk(int n,int a[],int b[]);

int op[20];

int k;

int main()

{

int n,a[20],b[20],i,c;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

scanf("%d",&b[i]);

}

fdbk(n,a,b);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void fdbk(int n,int a[],int b[])

{

int i;

k=0;

for(i=0;i<n;i=i+2)

{

if(a[i]==b[i])

{

if(a[i+1]>b[i+1])

{

op[k]=a[i];

op[k+1]=a[i+1];

k=k+2;

}

else

{

op[k]=b[i];

op[k+1]=b[i+1];

k=k+2;

}

}

else

{

op[k]=a[i];

op[k+1]=a[i+1];

k=k+2;

op[k]=b[i];

op[k+1]=b[i+1];

k=k+2;

}

}

}

**106. Store Consequtives**

Write a program to obtain a new array that contains the consequtive values of the given input array. The output array is named as output1.

Include a function named **storeConsequtives** that accepts 2 arguments and its return type is void. The first argument is the input array and the second argument is an int that corresponds to the size of the array . The output array is stored in a global variable named output1.

If the size of the array is negative or if any of the elements in the array are negative , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer array.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

4

2

5

1

4

**Sample Output 1:**

3

6

2

5

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

#include<stdio.h>

void cons(int n,int a[]);

int op[20];

int k,c;

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

cons(n,a);

c=k;

for(i=0;i<c;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void cons(int n,int a[])

{

int i;

k=0;

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

op[k]=a[i]+1;

k++;

}

}

**109. powerOfTwo**

Read the question carefully and follow the input and output format.  
  
Check whether given number is a power of 2 or not .If yes Print 'Yes' else 'No'  
  
Input and Output Format :  
Input consists of an integer number. And output is a single line that displays 'Yes' or 'No'  
  
Print "Number too small" if the number is less than 0  
Print "Number too large" if the number is greater than 32767  
  
Include a function named powerOfTwo(int n) that returns an integer.  
  
  
**Sample Input 1:**  
3  
**Sample Output 1:**  
No  
  
**Sample Input 2**:  
34569  
**Sample Output 2:**  
Number too large

#include<stdio.h>

#include<math.h>

int power(int n);

int main()

{

int n,c;

scanf("%d",&n);

c=power(n);

if(c==1)

{

printf("yes");

}

else

printf("no");

return 0;

}

int power(int n)

{

int i,flag;

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

{

flag=0;

if(pow(2,i)==n)

{

flag=1;

break;

}

}

if(flag==1)

return 1;

else

return 2;

}

**110. sumThreeLargest**

Read the question carefully and follow the input and output format.  
  
Write a program to find the sum of first ,second and third largest element in the given array.  
  
**Input and Output Format:**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements . Output consists of an Integer, the sum.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative numbers available in the input array and terminate the program.  
  
Include a function named sumThreeLargest(int array[], int n) whose return type is integer  
  
**Sample Input 1:**  
8  
1  
2  
2  
3  
4  
5  
5  
4  
  
**Sample Output 1:**  
12  
  
**Sample Input 2:**  
4  
1  
2  
-3  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int large(int a[],int n);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",large(a,n));

return 0;

}

int large(int a[],int n)

{

int i,j,op[20],k,sum,t;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]==a[j])

{

a[j]=-1;

}

}

}

k=0;

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

{

if(a[i]>0)

{

op[k]=a[i];

k++;

}

}

for(i=0;i<k;i++)

{

for(j=i+1;j<k;j++)

{

if(op[i]<op[j])

{

t=op[i];

op[i]=op[j];

op[j]=t;

}

}

}

sum=op[0]+op[1]+op[2];

return sum;

}

**111. Change using Minimal Coins / Notes**

Ram needs to pay the school fees of his 6 year old kid. As he is busy with his work, he is not finding time to go to the school to make payment. His kid's school doesn't accept online payment. So he decided to send the fee amount through his kid. The available denominations of rupees or coins is 500, 100, 50, 10, 5 and 1. Can you write a program to find the minimal number of coins or notes to be given to his kid?

Include a function named **countRupees** that accepts an integer and returns an integer that corresponds to the minimal number of coins/rupee notes. The function returns -1 if the input is negative.

**Input and Output Format:**

Input consists of a single integer that corresponds to the fee amount to be paid.

Output consists of an integer that corresponds to the minimal number of coins or rupee notes.

Print Invalid Input if the input value is negative.

**Sample Input 1:**

682

**Sample Output 1:**

8

**Sample Input 2:**

**-**2345

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int coin(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",coin(n));

return 0;

}

int coin(int n)

{

int amt1,amt2,amt3,amt4,amt5,amt6,sum;

amt1=n/500;

n=n%500;

amt2=n/100;

n=n%100;

amt3=n/50;

n=n%50;

amt4=n/10;

n=n%10;

amt5=n/5;

n=n%5;

amt6=n/1;

n=n%1;

sum=amt1+amt2+amt3+amt4+amt5+amt6;

return sum;

}

**115. primeFactorialSum**

Read the question carefully and follow the input and output format.  
  
In a given input number , find out the sum of factorial of digits that are prime.  
  
**Input and Output Format :**  
Input consists of an integer. Output consists of the factorial sum.  
1) Print "Number too large" when the given input number is greater than 32767  
2) Print "Number too small" when the given input number is a negative number.  
  
Include a function named primeFactorialSum(int number) whose return type is an integer.  
  
**Sample Input 1:**  
123  
**Sample Output 1:**  
8  
  
Hint : 2! + 3! = (8)  
  
**Sample Input 2:**  
32768  
**Sample Output 2:**  
Number too large

#include<stdio.h>

int prime(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",prime(n));

return 0;

}

int prime(int n)

{

int d,flag,i,j,fact,sum=0;

while(n>0)

{

flag=0;

d=n%10;

if(d==1)

{

flag=1;

}

for(i=2;i<d;i++)

{

if(d%i==0)

flag=1;

break;

}

if(flag==0)

{

fact=1;

for(j=1;j<=d;j++)

{

fact=fact\*j;

}

sum=sum+fact;

}

n=n/10;

}

return sum;

}

**117. sumTwoFive**

Read the question carefully and follow the input and output format.  
  
Given an integer array find the sum of elements which end with 2 or 5.  
  
**Input and Output Format:**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer, the sum.  
  
1) Print "Invalid array size" when size of the array is negative .  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named sumTwoFive(int array[], int size) whose return type is an integer, the sum  
  
**Sample Input 1:**  
5  
22  
35  
5  
2  
10  
  
**Sample Output 1:**  
64  
  
**Sample Input 2:**  
-6  
  
**Sample Output 2:**  
Invalid array size

#include<stdio.h>

int sum(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",sum(n,a));

return 0;

}

int sum(int n,int a[])

{

int i,d,sum=0;

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

{

d=a[i]%10;

if(d==2||d==5)

{

sum=sum+a[i];

}

}

return sum;

}

**118. sumPrimeArray**

Read the question carefully and follow the input and output format.  
  
John is working in a bank. He has created account details transaction in a file and protected it with a password. He sent the file to his manager for review. The file is protected with a password. The password is the sum of Prime numbers. Write a function to generate the password.  
  
**Input and Output Format:**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer, which is the sum.  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
3) Print 0, when there are no prime numbers in a given input array.  
  
Include a function named sumPrimeArray(int array[], int size) whose return type is an integer, which is the prime sum.  
  
**Sample Input 1:**  
5  
1  
2  
3  
4  
5  
  
**Sample Output 1:**  
10  
  
**Sample Input 2:**  
3  
4  
8  
9  
  
**Sample Output 2:**  
0

#include<stdio.h>

int prime(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",prime(n,a));

return 0;

}

int prime(int n,int a[])

{

int i,j,flag,sum=0;

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

{

flag=0;

if(a[i]==1)

{

flag=1;

}

for(j=2;j<a[i];j++)

{

if(a[i]%j==0)

{

flag=1;

break;

}

}

if(flag==0)

{

sum=sum+a[i];

}

}

return sum;

}

**119. LCM**

Write a program to calculate the LCM of the 2 given integers.

Include a function named **calculateLCM** that accepts 2 integer arguments and returns an int that corresponds to the LCM of the 2 numbers.

Print Invalid Input and terminate the program in the following cases:

1. Any of the 2 inputs is greater than 1000

2. Any of the 2 inputs is negative

**Input and Output Format:**

Input consists of 2 integers.

Output consists of a single integer that corresponds to the LCM.

**Sample Input 1:**

10

8

**Sample Output 1:**

40

**Sample Input 2:**

50000

**Sample Output 2:**

Invalid Input

#include<stdio.h>

int lcm(int a,int b);

int main()

{

int a,b;

scanf("%d",&a);

scanf("%d",&b);

printf("%d",lcm(a,b));

return 0;

}

int lcm(int a,int b)

{

int max,c;

if(a>b)

{

max=a;

}

else

max=b;

while(1)

{

if(max%a==0 && max%b==0)

{

c=max;

break;

}

else

max++;

}

return c;

}

**120. calculateBonus**

Read the question carefully and follow the input and output format.  
  
Given the basic salary as input, write a program to calculate the bonus and display it.  
  
The bonus will be calculated based on the below category.  
Basic>20000 bonus=17%of basic+1500  
Basic>15000 bonus=15%of basic+1200  
Basic>10000 bonus=12%of basic+1000  
for rest =8%of basic+500  
  
**Input and Output Format :**  
  
First line of input consists of n, the basic salary.  
Output is a single integer that displays the bonus.  
  
Print "Number too large" when the given input numbers is greater than 32767 .  
Print "Number too small" when the given input is a negative number.  
  
Include a function named calculateBonus(int basic) whose return type is an integer, the bonus.  
  
**Sample Input 1:**  
21000  
  
**Sample Output 1:**  
5070  
  
**Sample Input 2:**  
327678  
**Sample Output 2:**  
Number too large

#include<stdio.h>

int basic(int n);

int main()

{

int n;

scanf("%d",&n);

printf("%d",basic(n));

return 0;

}

int basic(int n)

{

int sum;

if(n>20000)

{

sum=0.17\*n+1500;

}

else if(n>15000)

{

sum=0.15\*n+1200;

}

else if(n>10000)

{

sum=0.12\*n+1000;

}

else

{

sum=0.08\*n+500;

}

return sum;

}

**123. highestProfitYear**

Read the question carefully and follow the input and output format.  
  
An array holds information of a company profit margin and year. Find out the year in which highest revenue was earned. Assume the first index of array indicates year and the next index indicates the amount of money earned by the company and so on.  
  
**Input and Output Format :**  
First line of input consists of n, the number of elements. Next n lines correspond to the array elements. Output consist of an integer, which is the sum  
  
1) Print "Invalid array size" when size of the array is a negative number and terminate the program.  
2) Print "Invalid input" when there is any negative number available in the input array and terminate the program.  
  
Include a function named highestProfitYear(int revenue[],int size) whose return type is an integer, which is the year.  
  
**Sample Input 1:**  
6  
2012  
10000  
2011  
5000  
2009  
4000  
  
**Sample Output 1:**  
2012  
  
**Sample Input 2:**  
8  
2015  
89745  
-2015  
  
**Sample Output 2:**  
Invalid input

#include<stdio.h>

int year(int n,int a[]);

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

printf("%d",year(n,a));

return 0;

}

int year(int n,int a[])

{

int i,c,max=0;

for(i=0;i<n;i=i+2)

{

if(a[i+1]>max)

{

max=a[i+1];

c=a[i];

}

}

return c;

}

**129. Descending Order Sort**

Write a program to sort the given array in descending order.

Include a function named **sortArray** that accepts 2 arguments and its return type is void. The first argument is the input array and the second argument is an int that corresponds to the size of the array .

If the size of the array is negative or if any of the elements in the array are negative , print “Invalid Input” and terminate the program.

**Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of an integer array.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

**Sample Input 1:**

8

1

6

3

5

8

10

4

9

**Sample Output 1:**

10

9

8

6

5

4

3

1

**Sample Input 2:**

-5

**Sample Output 2:**

Invalid Input

#include<stdio.h>

void descending(int n,int a[]);

int op[20];

int k;

int main()

{

int n,a[20],i;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

descending(n,a);

for(i=0;i<k;i++)

{

printf("%d\n",op[i]);

}

return 0;

}

void descending(int n,int a[])

{

int i,j,t;

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

{

for(j=i+1;j<n;j++)

{

if(a[i]<a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

k=0;

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

{

op[k]=a[i];

k++;

}  
}