TIME COMPLEXITY

1.

Input:
A positive integer n
Output:
Print the value of the counter variable

For example:

Input	Result
9	12

Answer: (penalty regime: 0 %)

```
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int i=1;
    int p=1;
    int counter=2;
    while(p<=n)
    {
        i++;
        p+=i;
        counter+=3;
    }
    counter++;
    printf("%d\n", counter);
    return 0;
}
```

	Input	Expected	Got	
✔	9	12	12	✔
✔	4	6	6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.
Input:
A positive integer n
Output:
Print the value of the counter variable

Answer: (penalty regime: 0 %)

```
#include<stdio.h>
int main()
{
    int count=0;
    int n;
  scanf("%d",&n);
  if(n==1)
  {  count++;
   printf("*");
  }
  else
  { count++;
   for(int i=1; i<=n; i++)
   { count++;
    for(int j=1; j<=n; j++)
    {
        count++;
    count++;
    count++;
    break;
    } count++;
   } count++;
  }
 printf("%d",count);
 }
```

	Input	Expected	Got	
✔	3	33	33	✔
✔	1000	9003	9003	✔
✔	343	7317	7317	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.


Convert the following algorithm into a program and find its time complexity using counter method.
factor(num) {
for (i = 1; i <= num;++i)
{
if (num % i== 0)
{
printf("%d ", i);
}
}
}
Note: no need of counter increment for declarations and scanf() and counter variable printf() statement.
Input:
A positive integer n
Output:
Print the value of the counter variable
Answer:
#include<stdio.h>
int factor(int num){
int count=0;
for(int i=1; i<=num; ++i){
count++;
count++;
if(num%i==0){
//printf("%d",i);//
count++;
}
}
count++;
return count;
}
int main(){
int n;
scanf("%d",&n);
printf("%d",factor(n));
}
Input
Expected
Got

4.

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Convert the following algorithm into a program and find its time

complexity using counter method.

```
void function(int n)
{
   int i= 0;
   for(int i=n/2; i<n; i++)
      for(int j=1; j<n; j = 2 * j)
         for(int k=1; k<=( n * n ); 
            ++i;
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:
 A positive integer n
Output:
Print the value of the counter variable

Answer:

```
#include<stdio.h>
int function(int n){
    int count=0;
    int a=0;
    count++;
    for(int i=n/2;i<n;i++){
        count++;
        for(int j=1;j<n;j=2*j){
            count++;
            for(int k=1;k<n;k=k*2){
                count++;
                a++;
                count++;
            }
            count++;
        }
        count++;
    }
    count++;
    return count;
}
int main(){
    int b;
    scanf("%d",&b);
    printf("%d",function(b));
}
```

	Input	Expected	Got	
✔	4	30	30	✔
✔	10	212	212	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

5.

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Convert the following algorithm into a program and find its time complexity using counter method.

```
void reverse(int n)
{
   int rev = 0, remainder;
   while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n/= 10;

    }
printf(rev);
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:
 A positive integer n
Output:
Print the value of the counter variable

Answer:

Answer:

```c
#include<stdio.h>
int reverse(int n){
    int count=0;
    int rev=0;
    int rem;
    count++;
    while(n!=0){
        count++;
        rem=n%10;
        count++;
        rev=rev*10+rem;
        count++;
        n/=10;
        count++;
    }
    count++;
    count++;
    return count;
}
int main(){
    int n;
    scanf("%d",&n);
    printf("%d",reverse(n));
}
```

	Input	Expected	Got	
✔	12	11	11	✔
✔	1234	19	19	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

GREEDY METHOD

1.

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Write a program to take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin

Output Format:

print the integer which is change of the number.

Example Input :

58

Output:

4

Explanation:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
int min(int a) {
  int b[]={1000, 500, 100, 50, 20, 10, 5, 2, 1};
  int n=sizeof(b)/sizeof(b[0]);
  int count=0;
  for (int i=0; i<n; i++) {
    if (a==0) {
      break;
    }
    count+=a/b[i];
    a=a%b[i];
  }
  return count;
}
int main() {
  int a;
  scanf("%d", &a);
  int result=min(a);
  printf("%d\n", result);
  return 0;
}
```

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.Cookies Problem

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

Example 1:

Input:

3

1 2 3

2

1 1

Output:

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

Constraints:

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
int main(){
    int n1,n2,count=0,ptr1=0,ptr2=0;
    scanf("%d",&n1);
    int children[n1];
    for(int i=0; i<n1; i++) scanf("%d", &children[i]);
    scanf("%d", &n2);
    int cookies[n2];
    for(int i=0; i<n2; i++) scanf("%d", &cookies[i]);
    while(ptr1 < n1 && ptr2 < n2){
        if(cookies[ptr1] >= children[ptr2]){
            count++;
            ptr1++;
            ptr2++;
        }
    }
    printf("%d",count);
    return 0;
}
```

	Input	Expected	Got	
✔	2 1 2 3 1 1 1	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.Burger Problem

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories. If he has eaten i burgers with c calories each, then he has to run at least $i^2 * c$ kilometers to burn out the calories. For example, if he ate 3 burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$. But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

Input Format

First Line contains the number of burgers

Second line contains calories of each burger which is a space-separate integers

Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

Sample Input

```
3
5 10 7
```

Sample Output

```
76
```

For example:

Test	Input	Result
Test Case 1	3 1 1 2	10

Answer: (penalty regime: 0 %)

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int comp(const void*A, const void *B){
    return (*(int*)B)-(*(int*)A);
}
int main(){
    int n,km=0;
    scanf("%d", &n);
    int cal[n];
    for(int i=0; i<n; i++){
        scanf("%d", &cal[i]);
    }
    qsort(cal,n,sizeof(int),comp);
    for(int i=0; i<n; i++){
        km += (pow(n,i)*cal[i]);
    }
    printf("%d",km);
    return 0;
}
```

	Test	Input	Expected	Got	
✔	Test Case 1	3 1 1 2	10	10	✔
✔	Test Case 2	4 7 4 9 6	888	888	✔
✔	Test Case 3	3 5 10 7	76	76	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.Array sum Max problem

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Given an array of N integer, we have to maximize the sum of arr[i] * i, where i is the index of the element (i = 0, 1, 2, ... N).Write an algorithm based on Greedy technique with a Complexity O(nlogn).

Input Format:

First line specifies the number of elements-n

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

5

2 3 5 4 0

Sample output:

40

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
#include <stdlib.h>
int comp(const void *a, const void *b){
    return (*(int*)a)-(*(int*)b);
}
int main(){
    int n,sum=0;
    scanf("%d", &n);
    int arr[n];
    for(int i=0;i<n; i++){
        scanf("%d", &arr[i]);
    }
    qsort(arr,n,sizeof(int),comp);
    for(int i=0; i<n; i++){
        sum += arr[i]*i;
    }
    printf("%d", sum);
    return 0;
}
```

	Input	Expected	Got	
✔	6 2 5 7 3 4	60	60	✔
✔	10 1 2 3 4 4 4 4 5 5 5	101	101	✔
✔	3 10 3	10	10	✔

Passed all tests! ✔

Correct

5. GREEDY METHOD

Question 1

Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs(1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

For example:

Input	Result
7 1 2 3 4 5 6	28

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
#include <stdlib.h>
int comp1(const void *a, const void *b){
    return (*(int*)a)-(*(int*)b);
}
int comp2(const void *a, const void *b){
    return (*(int*)b)-(*(int*)a);
}

int main(){
    int n,sum=0;
    scanf("%d",&n);
    int arr1[n],arr2[n];
    for(int i=0; i<n; i++){
        scanf("%d", &arr1[i]);
    }
    for(int i=0; i<n; i++){
        scanf("%d", &arr2[i]);
    }
    qsort(arr1,n,sizeof(int),comp1);
    qsort(arr2,n,sizeof(int),comp2);
    for(int i=0; i<n; i++){
        sum += arr1[i]*arr2[i];
    }
    printf("%d", sum);
    return 0;
}
```

	Input	Expected	Got	
✔	7 1 2 3 4 5 6	28	28	✔
✔	4 7 5 1 2 1 3 4 1	22	22	✔
✔	5 10 10 10 10 40 8 9 4 3 10	990	990	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.