Recursion Questions

1. Predict output of following program

```
#include <stdio.h>

int fun(int n)
{
   if (n == 4)
      return n;
   else return 2 * fun ( n+1 );
}

int main()
{
  printf("%d ", fun(2));
  return 0;
}
```

Options
A.4
B. 8
C. 16
D.Runtime Error

Explaination:
Fun(2) = 2 * Fun(3) and Fun(3) = 2 * Fun(4) ....(i)
Fun(4) = 4 ......(ii)
From equation (i) and (ii),
Fun(2) = 2 * 2 * Fun(4)
Fun(2) = 2 * 2 * 4
Fun(2) = 16.

So, C is the correct answer

2. Consider the following recursive function fun(x, y). What is the value of fun(4, 3)

```
int fun(int x, int y)
{
  if (x == 0)
    return y;
  return fun(x - 1,  x + y);
}
```

Options
A.13
B. 8

C. 9
D.Runtime Error

Explaination:
The function fun() calculates and returns ((1 + 2 … + x-1 + x) +y) which is x(x+1)/2 + y.

So, A is the correct answer

3.What does the following function print for n = 25?

```
void fun(int n)
{
  if (n == 0)
    return;

  printf("%d", n%2);
  fun(n/2);
}
```

Options
A.11001
B. 10011
C. 11111
D.00000

Explaination:
The function mainly prints binary representation in reverse order.

So, B is the correct answer

4.What does the following function do?

```
int fun(int x, int y)
{
    if (y == 0)   return 0;
    return (x + fun(x, y-1));
}
```

Options
A. x+y
B. x+x*y
C. x*y
D. x^y

Explaination:
The function adds x to itself y times which is x*y.

So, C is the correct answer

5.What does fun2() do in general?

```
int fun(int x, int y)
{
   if (y == 0)   return 0;
   return (x + fun(x, y-1));
}

int fun2(int a, int b)
{
   if (b == 0) return 1;
   return fun(a, fun2(a, b-1));
}
```

Options
A. x^y
B. x+x*y
C. x*y
D. x^y

Explaination:
The function multiplies x to itself y times which is x^y.

So, A is the correct answer

6.Output of following program?

```
#include<stdio.h>
void print(int n)
{
   if (n > 4000)
      return;
   printf("%d ", n);
   print(2*n);
   printf("%d ", n);
}

int main()
{
   print(1000);
   getchar();
   return 0;
}
```

Options

A. 1000 2000 4000
B. 1000 2000 4000 4000 2000 1000
C. 1000 2000 4000 2000 1000
D. 1000 2000 2000 1000

Explaination:
First time n=1000 Then 1000 is printed by first printf function then call print(2*1000) then again print 2000 by printf function then call print(2*2000) and it prints 4000 next time print(4000*2) is called. Here 8000 is greater than 4000 condition becomes true and it return at function(2*4000). Here n=4000 then 4000 will again print through second printf. Similarly print(2*2000) after that n=2000 then 2000 will print and come back at print(2*1000) here n=1000, so print 1000 through second printf. Option (B) is correct.

So, B is the correct answer

7.What does the following function do?

```
int fun(unsigned int n)
{
   if (n == 0 || n == 1)
      return n;

   if (n%3 != 0)
      return 0;

   return fun(n/3);
}
```

Options
A. It returns 1 when n is a multiple of 3, otherwise returns 0
B. It returns 1 when n is a power of 3, otherwise returns 0
C. It returns 0 when n is a multiple of 3, otherwise returns 1
D. It returns 0 when n is a power of 3, otherwise returns 1

Explaination:

Lets solve with example, n = 27 which power of 3. First time if condition is false as n is neither equal to 0 nor equal to 1 then 27%3 = 0. Here, again if condition false because it is equal to 0. Then fun(27/3) will call. Second time if condition is false as n is neither equal to 0 nor equal to 1 then 9%3 = 0. Here again if condition false because it is equal to 0. Then fun (9/3) will call and third time if condition is false as n is neither equal to 0 nor equal to 1 then 3%3 = 0. Here again if condition false because it is equal to 0. Then fun(3/3) will call here n==1 if condition gets true and it return n i.e. 1. Option (B) is correct.

So, B is the correct answer

8.Predict the output of following program

```
#include <stdio.h>
int f(int n){
   if(n <= 1)
      return 1;
   if(n%2 == 0)
      return f(n/2);
   return f(n/2) + f(n/2+1);
}
int main()
{
   printf("%d", f(11));
   return 0;
}
```

Options
A. Stack Overflow
B. 3
C. 4
D. 5

Explaination:

On successive recursion F(11) will be decomposed into F(5) + F(6) -> F(2) + F(3) + F(3) -> F(1) + 2 * [F(1) + F(2)] -> 1 + 2 * [1 + F(1)] -> 1 + 2 * (1 + 1) -> 5. Hence , option D is the correct answer i.e, 5.

So, D is the correct answer

9.Consider the following recursive C function that takes two arguments

```
unsigned int foo(unsigned int n, unsigned int r) {
  if (n  > 0) return (n%r +  foo (n/r, r ));
  else return 0;
}
```

What is the return value of the function foo when it is called as foo(345, 10) ?
Options
A. 345
B. 12
C. 5
D. 3

Explaination:

The call foo(345, 10) returns sum of decimal digits (because r is 10) in the number n. Sum of digits for 345 is 3 + 4 + 5 = 12.

So, B is the correct answer

10.Consider the same recursive C function that takes two arguments

```
unsigned int foo(unsigned int n, unsigned int r) {
  if (n  > 0) return (n%r +  foo (n/r, r ));
  else return 0;
}
```

What is the return value of the function foo when it is called as foo(513, 2)?

Options

A. 9
B. 8
C. 5
D. 2

Explaination:

foo(513, 2) will return 1 + foo(256, 2). All subsequent recursive calls (including foo(256, 2)) will return 0 + foo(n/2, 2) except the last call foo(1, 2) . The last call foo(1, 2) returns 1. So, the value returned by foo(513, 2) is 1 + 0 + 0…. + 0 + 1. The function foo(n, 2) basically returns sum of bits (or count of set bits) in the number n.

So, D is the correct answer

11.Predict the output of following program

```
#include<stdio.h>
int f(int *a, int n)
{
  if(n <= 0) return 0;
  else if(*a % 2 == 0) return *a + f(a+1, n-1);
  else return *a - f(a+1, n-1);
}

int main()
{
  int a[] = {12, 7, 13, 4, 11, 6};
  printf("%d", f(a, 6));
  getchar();
  return 0;
}
```

Options

A. -9
B. 5
C. 15
D. 19

Explaination:
f() is a recursive function which adds f(a+1, n-1) to *a if *a is even. If *a is odd then f() subtracts f(a+1, n-1) from *a. See below recursion tree for execution of f(a, 6). .

So, C is the correct answer

12.Output of following program?

```
#include <stdio.h>
int fun(int n, int *f_p)
{
   int t, f;
   if (n <= 1)
   {
      *f_p = 1;
      return 1;
   }
   t = fun(n- 1,f_p);
   f = t+ * f_p;
   *f_p = t;
   return f;
}

int main()
{
   int x = 15;
   printf (" %d n", fun(5, &x));
   return 0;
}
```

Options
A. 6
B. 8
C. 14
D. 15

Explaination:
Let x is stored at location 2468 i.e. &x = 2468

(dots are use just to ensure alignment)
x = 15
fun(5, 2468)

...{t = fun(4, 2468)
.......{t = fun(3, 2468)
...........{t = fun(2,2468)
...............{t = fun(1, 2468)
...................{// x=1
........................return 1}
................t = 1
................f = 2 //1+1 //since *f_p is x
................x = t = 1
................return 2}
...........t = 2
...........f = 2+1
...........x = t = 2
...........return 3}
........t = 3
........f = 3+2
........x = t = 3
........return 5}
....t = 5
....f = 5+3
....x = t = 5
....return 8}

which implies fun (5,2468) is 8.

So, B is the correct answer

13.Consider the following function

Give a value q (to 2 decimals) such that f(q) will return q:_____.

```
double f(double x){
  if (abs(x*x - 3) < 0.01) return x;
  else return f(x/2 + 1.5/x);
}
```

Options
A. 1.73
B. 2.24
C. 4.22
D. 3.42

A is the correct answer

14.Consider the following function

Which one of the following is TRUE?

```
int f(int j)
{
  static int i = 50;
  int k;
  if (i == j)
  {
    printf("something");
    k = f(i);
    return 0;
  }
  else return 0;
}
```

Options
A. The function returns 0 for all values of j.
B. The function prints the string something for all values of j.
C. The function returns 0 when j = 50
D.The function will exhaust the runtime stack or run into an infinite loop when j = 50

Explaination:
When j is 50, the function would call itself again and again as neither i nor j is changed inside the recursion.

So, D is the correct answer

15.Consider the following Program

Which one of the following is Correct Output?

```
#include<stdio.h>
void crazy(int n,int a,int b)
{
   if (n <= 0)  return;
   crazy(n-1, a, b+n);
   printf("%d %d %dn",n,a,b);
   crazy(n-1, b, a+n);
}

int main()
{
   crazy(3,4,5);
   return 0;
}
```

Options

A.
1 4 10
2 4 8
1 8 6
3 4 5
1 5 9
2 5 7
1 7 7
B.
3 4 5
1 4 10
2 4 8
1 8 6
1 5 9
2 5 7
1 7 7

C.
1 4 10
2 4 8
1 8 6
3 4 5

D.
3 4 5
1 5 9
2 5 7
1 7 7

A is the correct answer

16.Consider the following Function

```
int f(int n)
{
  static int i = 1;
  if (n >= 5)
    return n;
  n = n+i;
  i++;
  return f(n);
}
```

The value returned by f(1) is

Options
A. 5

B. 6
C. 7
D. 8

Explaination:
See Question 3 of http://www.geeksforgeeks.org/c-language-set-2/

So, C is the correct answer

17.Consider the following C function.

```
int fun (int n)
{
  int x=1, k;
  if (n==1) return x;
  for (k=1; k<n; ++k)
     x = x + fun(k) * fun(n – k);
  return x;
}
```

The return value of fun(5) is __________.

Options
A. 0
B. 26
C.51
D.71

Explaination:
fun(5) = 1 + fun(1) * fun(4) + fun(2) * fun(3) +
            fun(3) * fun(2) + fun(4) * fun(1)
       = 1 + 2*[fun(1)*fun(4) + fun(2)*fun(3)]

Substituting fun(1) = 1
       = 1 + 2*[fun(4) + fun(2)*fun(3)]

Calculating fun(2), fun(3) and fun(4)
fun(2) = 1 + fun(1)*fun(1) = 1 + 1*1 = 2
fun(3) = 1 + 2*fun(1)*fun(2) = 1 + 2*1*2 = 5
fun(4) = 1 + 2*fun(1)*fun(3) + fun(2)*fun(2)
       = 1 + 2*1*5 + 2*2 = 15

Substituting values of fun(2), fun(3) and fun(4)
fun(5) = 1 + 2*[15 + 2*5] = 51

So, C is the correct answer

18.Consider the following recursive C function. If get(6) function is being called in main() then how many times will the get() function be invoked before returning to the main()?

```
void get (int n)
{
  if (n < 1) return;
  get(n-1);
  get(n-3);
  printf("%d", n);
}
```

Options
A. 15
B. 25
C.35
D.45

Explaination:

```
get(6) [25 Calls]
                       /     \
            [17 Calls] get(5)      get(3) [7 Calls]
                   /    \
                get(4)    get(2)[5 Calls]
               /   \
    [7 Calls] get(3)  get(1)[3 Calls]
             /    \
          get(2)   get(0)
         /   \
[3 Calls]get(1)  get(-1)
  / \
get(0) get(-2)
```

We can verify the same by running below program. [sourcecode language="CPP"] # include int count = 0; void get (int n) { count++; if (n < 1) return; get(n-1); get(n-3); } int main() { get(6); printf("%d ", count); } [/sourcecode] Output: 25

So, B is the correct answer

19.What will be the output of the following C program?

```
void count(int n)
{
   static int d = 1;
```

```
    printf("%d ", n);
    printf("%d ", d);
    d++;
    if(n > 1) count(n-1);
    printf("%d ", d);
}
int main()
{
    count(3);
}
```

Options
A. 3 1 2 2 1 3 4 4 4
B. 3 1 2 1 1 1 2 2 2
C.3 1 2 2 1 3 4
D.3 1 2 1 1 1 2

Explaination:
count(3) will print value of n and d. So 3 1 will be printed and d will become 2.

Then count(2) will be called. It will print value of n and d. So 2 2 will be printed and d will become 3.

Then count(1) will be called. It will print value of n and d. So 1 3 will be printed and d will become 4.

Now count(1) will print value of d which is 4. count(1) will finish its execution.

Then count(2) will print value of d which is 4.

Similarly, count(3) will print value of d which is 4.
So series will be A.

So, A is the correct answer

20.The function f is defined as follows:

```
int f (int n) {
    if (n <= 1) return 1;
    else if (n % 2  ==  0) return f(n/2);
    else return f(3n - 1);
}
```

Assuming that arbitrarily large integers can be passed as a parameter to the function, consider the following statements.
1. The function f terminates for finitely many different values of n ≥ 1.
ii. The function f terminates for infinitely many different values of n ≥ 1.
iii. The function f does not terminate for finitely many different values of n ≥ 1.
iv. The function f does not terminate for infinitely many different values of n ≥ 1.
Which one of the following options is true of the above?

Options
A. (i) and (iii)
B. (i) and (iv)
C. (ii) and (iii)
D. (ii) and (iv)

Explaination:
The function terminates for all values having a factor of 2 {(2.x)2==0}
So, (i) is false and (ii) is TRUE.
Let n = 3, it will terminate in 2nd iteration.
Let n=5, it will go like 5 - 14 - 7 - 20 - 10 - 5 – and now it will repeat.
And any number with a factor of 5 and 2, there are infinite recursions possible.
So, (iv) is TRUE and (iii) is false.

So, D is the correct answer

1. Print Decreasing

1. You are given a positive number n.
2. You are required to print the counting from n to 1.
3. You are required to not use any loops.

Examples:

Input : 5
Output :

```
5
4
3
2
1
```

2. Print Increasing

1. You are given a positive number n.
2. You are required to print the counting from 1 to n.
3. You are required to not use any loops.

Examples:

Input : 5
Output :

```
1
2
3
4
5
```

3. Print Increasing Decreasing

1. You are given a positive number n.
2. You are required to print the counting from n to 1 and back to n again
3. You are required to not use any loops.

Examples:

Input : 5
Output :

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

4. Factorial

1. You are given a positive number n.
2. You are required to calculate the factorial of the number.
3. You are required to not use any loops.

Examples:

Input : 5
Output : 120

5. Power-linear

1. You are given a number x.
2. You are given another number n.
3. You are required to calculate x raised to the power n.

Examples:

Input : 2 5
Output : 32

6. Power-logarithmic

1. You are given a number x.
2. You are given another number n.
3. You are required to calculate x raised to the power n.

Examples:

Input : 2 5
Output : 32

7. Print Zigzag

1. Here are a few sets of inputs and outputs for your reference
Input1 -> 1
Output1 -> 1 1 1

Input2 -> 2
Output2 -> 2 1 1 1 2 1 1 1 2

Examples:

Input : 3
Output : 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

8. Array Insertion

1. You are given a Empty Array of size N.
2. You have to Insert Elements in Array.

Examples:

Input:
arr[N]

Output:

1 2 3 4 5

9. Print Array Elements

1. You are given a Array of size N.
2. You have to Print all the Elements of Array.

Examples:

Input:
arr[N] = {1,2,3,4,5}

Output:

1 2 3 4 5

10. Print Array Size

1. You are given a Array.
2. You have to return size of array.

Examples:

Input:
arr[N] = {1,2,3,4,5}

Output:

5

11. Print array elements in reverse order

1. You are given a Array.
2. You have to print array elements in reverse order.

Examples:

Input:
arr[N] = {1,2,3,4,5}

Output:

5 4 3 2 1

12. Find the kth element of Array

1. You are given a Array.
2. You have to print kth array elements .

Examples:

Input:
arr[N] = {1,2,3,4,5}
k = 3

Output:

4

14. Find the first occurence of a element Array

1. You are given a Array.
2. You have to return the first index of given array elements .

Examples:

Input:
arr[N] = {1,2,3,4,5, 3}
value = 3

Output:

2

15. Find the Last occurence of a element Array

1. You are given a Array.
2. You have to return the last index of given array elements .

Examples:

Input:
arr[N] = {1,2,3,4,5,3}
value = 3

Output:

5

16. Given an array, compute recursively the number of times that the value 5 appears in the array

Examples:

Input:
arr[N] = {1,2,3,4,5,3,5,5,5}
value = 3

Output:

4

17. Search a Value in Array recursively

Examples:

Input:
arr[N] = {23, 12, 4, 65, 67}
value = 4

Output:

Yes at index 2

18. find all indices of a number

Examples:

Input:
arr[N] = {1, 2, 3, 2, 2, 5}
x = 2

Output:

1 3 4

19. Recursive Implementation of atoi()

The **atoi()** function takes a string (which represents an integer) as an argument and returns its value.

Examples:

Input:
char str[] = "112";

Output:

112

20. Check if a number is Palindrome

Given an integer, write a function that returns true if the given number is palindrome, else false. For example, 12321 is palindrome, but 1451 is not palindrome.

Examples:

Input:
int = 11211;

Output:

Palindrom

Input:
int = 12345;

Output:

Not a Palindrom

21. Write a recursive function that, given a string s, prints the characters of s in reverse order.

Examples:

Input:
char ch[] = “hello”

Output:

olleh

22. Function to copy string (Recursive)

Given two strings, copy one string to other using recursion. We basically need to write our own recursive version of strcpy in C/C++

Examples:

Input:
s1 = "hello"
s2 = "geeksforgeeks"

Output:

s2 = "hello"

23. Count Digits of a Number

Examples:

Input:
12345

Output:

5

24. Length Of a String

Examples:

Input:
string str = "abcdefghijklmnopqrstuvwxyz";

Output:

26

25. Sum of Digits

Examples:

Input:
12345

Output:

15

26. Print N Fabonacci Series

Examples:

Input:
N = 10

Output:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

27. Tower Of Hanoi

1. There are 3 towers. Tower 1 has n disks, where n is a positive number. Tower 2 and 3 are empty.
2. The disks are increasingly placed in terms of size such that the smallest disk is on top and largest disk is at bottom.
3. You are required to
    3.1. Print the instructions to move the disks.
    3.2. from tower 1 to tower 2 using tower 3
    3.3. following the rules
        3.3.1 move 1 disk at a time.
        3.3.2 never place a smaller disk under a larger disk.
        3.3.3 you can only move a disk at the top.

Examples:

Input :
3
10
11
12
Output :

```
1[10 -> 11]

2[10 -> 12]

1[11 -> 12]

3[10 -> 11]

1[12 -> 10]

2[12 -> 11]

1[10 -> 11]
```

27. Josephus Problem

1. You are given two numbers N and K.
2. N represents the total number of soldiers standing in a circle having position marked from 0 to N-1.
3. A cruel king wants to execute them but in a different way.
4. He starts executing soldiers from 0th position and proceeds around the circle in clockwise direction.
5. In each step, k-1 soldiers are skipped and the k-th soldier is executed.
6. The elimination proceeds around the circle (which is becoming smaller and smaller as the executed soldiers are removed), until only the last soldier remains, who is given freedom.
7. You have to find the position of that lucky soldier.

Examples:

Input :
4
2
Output :

```
0
```

28. Generating subarrays using recursion

Given an array, generate all the possible subarrays of the given array using recursion.

Examples:

Input :
[1, 2, 3]

Output :

[1], [1, 2], [2], [1, 2, 3], [2, 3], [3]

29. Print all subsequences of a string

Given a string, we have to find out all subsequences of it. A String is a subsequence of a given String, that is generated by deleting some character of a given string without changing its order.

Examples:

Input :
abc

Output :

a, b, c, ab, bc, ac, abc

30. Program to print all substrings of a given string

Given a string as an input. We need to write a program that will print all non-empty substrings of that given string.

Examples:

Input :
abcd

Output :

a b c d ab bc cd abc bcd abcd

31. Given a string, print all possible palindromic partitions

Given a string, find all possible palindromic partitions of given string.

Examples:

```
Input:  nitin
Output: n i t i n
        n iti n
        nitin

Input:  geeks
Output: g e e k s
        g ee k s
```

32. Print all possible strings of length k that can be formed from a set of n characters

Given a set of characters and a positive integer k, print all possible strings of length k that can be formed from the given set.

Examples:

Input :
set[] = {'a', 'b'}, k = 3

Output :

aaa aab aba abb baa bab bba bbb

33. Find all even length binary sequences with same sum of first and second half bits

Given a number n, find all binary sequences of length 2n such that sum of first n bits is same as sum of last n bits.

Examples:

Input :
N = 2

Output :

0101 1111 1001 0110 0000 1010

Input :
N = 3

Output :

```
011011 001001 011101 010001 101011 111111
110011 101101 100001 110101 001010 011110
010010 001100 000000 010100 101110 100010
110110 100100
```

34. Print all possible expressions that evaluate to a target

Given a string that contains only digits from 0 to 9, and an integer value, target. Find out how many expressions are possible which evaluate to target using binary operator +, – and * in given string of digits.

Examples:

Input :
"123", Target : 6

Output :

{“1+2+3”, “1*2*3”}

Input :
“125”, Target : 7

Output :

{“1*2+5”, “12-5”}

35. String with additive sequence

Given a string, the task is to find whether it contains an additive sequence or not. A string contains an additive sequence if its digits can make a sequence of numbers in which every number is addition of previous two numbers. A valid string should contain at least three digit to make one additive sequence.

Examples:

Input :
s = “235813”

Output :

true

2 + 3 = 5, 3 + 5 = 8, 5 + 8 = 13

Input :
s = “12345678”
Output : false

36. Generate all binary strings without consecutive 1’s

Given an integer K. Task is Print All binary string of size K (Given number).

Examples:

Input : K = 3
Output : 000 , 001 , 010 , 100 , 101

Input : K = 4
Output :0000 0001 0010 0100 0101 1000 1001 1010

37. Recursive solution to count substrings with same first and last characters

We are given a string S, we need to find count of all contiguous substrings starting and ending with same character.

Examples:

Input : S = "abcab"
Output : 7
There are 15 substrings of "abcab"
a, ab, abc, abca, abcab, b, bc, bca
bcab, c, ca, cab, a, ab, b
Out of the above substrings, there
are 7 substrings : a, abca, b, bcab,
c, a and b.

Input : S = "aba"
Output : 4
The substrings are a, b, a and aba

38. All possible binary numbers of length n with equal sum in both halves

Given a number n, we need to print all n-digit binary numbers with equal sum in left and right halves. If n is odd, then mid element can be either 0 or 1.

Examples:
Input : n = 4
Output : 0000 0101 0110 1001 1010 1111
Input : n = 5
Output : 00000 00100 01001 01101 01010 01110 10001 10101 10010 10110 11011
11111

39. Combinations in a String of Digits

Given an input string of numbers, find all combinations of numbers that can be formed using digits in the same order.

Examples:

```
Input : 123
Output :1 2 3
        1 23
        12 3
        123

Input : 1234
Output : 1 2 3 4
        1 2 34
        1 23 4
        1 234
        12 3 4
        12 34
        123 4
        1234
```

40. Count consonants in a string (Iterative and recursive methods)

Given a string, count total number of consonants in it. A consonant is a English alphabet character that is not vowel (a, e, i, o and u). Examples of constants are b, c, d, f, g, ..

Examples:

```
Input : abc de
Output : 3
There are three consonants b, c and d.

Input : geeksforgeeks portal
Output : 12
```

41. First uppercase letter in a string (Iterative and Recursive)

Given a string find its first uppercase letter

Examples:

```
Input : skillBout
Output : B
```

42. Partition given string in such manner that i’th substring is sum of (i-1)’th and (i-2)’th substring

Partition given string in such manner that i’th substring is sum of (i-1)’th and (i-2)’nd substring.

Examples:
Input : "11235813"
Output : ["1", "1", "2", "3", "5", "8", "13"]

Input : "1111223"
Output : ["1", "11", "12", "23"]

Input : "1111213"
Output : ["11", "1", "12", "13"]

Input : "11121114"
Output : []

43. Power Set in Lexicographic order

This article is about generating Power set in lexicographical order.

Examples:

Input : abc
Output : a ab abc ac b bc c

44. Print all possible combinations of r elements in a given array of size n

Given an array of size n, generate and print all possible combinations of r elements in array. For example, if input array is {1, 2, 3, 4} and r is 2, then output should be {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4} and {3, 4}.
Following are two methods to do this.

45. Replace every array element with the product of every other element without using a division operator

Examples:

Input: { 1, 2, 3, 4, 5 }
Output: { 120, 60, 40, 30, 24 }

Input: { 5, 3, 4, 2, 6, 8 }
Output: { 1152, 1920, 1440, 2880, 960, 720 }

46. Find all distinct combinations of a given length

Given an integer array, find all distinct combinations of gien length k.

Examples:

Input: {1, 2, 3}, k = 2
Output: {1, 2}, {1, 3}, {2, 3}

Input: {1, 2, 1}, k = 2
Output: {1, 1}, {1, 2}

47. Longest Increasing Subsequence

Given an integer array, find all distinct combinations of gien length k.

Examples:

Input: arr[] = {3, 10, 2, 1, 20}
Output: Length of LIS = 3
The longest increasing subsequence is 3, 10, 20

Input: arr[] = {3, 2}
Output: Length of LIS = 1
The longest increasing subsequences are {3} and {2}

Input: arr[] = {50, 3, 10, 7, 40, 80}
Output: Length of LIS = 4
The longest increasing subsequence is {3, 7, 40, 80}

48. Find a triplet that sum to a given value

Given an array and a value, find if there is a triplet in array whose sum is equal to the given value. If there is such a triplet present in array, then print the triplet and return true. Else return false.

Examples:

Input: array = {12, 3, 4, 1, 6, 9}, sum = 24;
Output: 12, 3, 9
Explanation: There is a triplet (12, 3 and 9) present
in the array whose sum is 24.
Input: array = {1, 2, 3, 4, 5}, sum = 9
Output: 5, 3, 1
Explanation: There is a triplet (5, 3 and 1) present
in the array whose sum is 9.

49. Print all n-digit numbers whose sum of digits equals to given sum

Given number of digits n, print all n-digit numbers whose sum of digits adds upto given sum. Solution should not consider leading 0's as digits.

Examples:

```
Input:  N = 2, Sum = 3
Output:  12 21 30

Input:  N = 3, Sum = 6
Output:  105 114 123 132 141 150 204
       213 222 231 240 303 312 321
       330 402 411 420 501 510 600

Input:  N = 4, Sum = 3
Output:  1002 1011 1020 1101 1110 1200
       2001 2010 2100 3000
```

50. Maximum product subset of an array

Given an array a, we have to find maximum product possible with the subset of elements present in the array. The maximum product can be single element also.

Examples:

```
Input: a[] = { -1, -1, -2, 4, 3 }
Output: 24
Explanation : Maximum product will be ( -2 * -1 * 4 * 3 ) = 24

Input: a[] = { -1, 0 }
Output: 0
Explanation: 0(single element) is maximum product possible

Input: a[] = { 0, 0, 0 }
Output: 0
```

51. Quickselect Algorithm

Quickselect is a selection algorithm to find the k-th smallest element in an unordered list. It is related to the quick sort sorting algorithm.

Examples:

```
Input: arr[] = {7, 10, 4, 3, 20, 15}
        k = 3
Output: 7

Input: arr[] = {7, 10, 4, 3, 20, 15}
```

```
      k = 4
Output: 10
```

52. Find four elements that sum to a given value

Given an array of integers, find anyone combination of four elements in the array whose sum is equal to a given value X.

Examples:

```
Input: array = {10, 2, 3, 4, 5, 9, 7, 8}
    X = 23
Output: 3 5 7 8
Sum of output is equal to 23,
i.e. 3 + 5 + 7 + 8 = 23.

Input: array = {1, 2, 3, 4, 5, 9, 7, 8}
    X = 16
Output: 1 3 5 7
Sum of output is equal to 16,
i.e. 1 + 3 + 5 + 7 = 16.
```

53. Find all combinations that add upto given number

Given a positive number, find out all combinations of positive numbers that adds upto that number. The program should print only combinations, not permutations. For example, for input 3, either 1, 2 or 2, 1 should be printed.

Examples:

```
Input: N = 3
Output:
1 1 1
1 2
3

Input: N = 5
Output:
1 1 1 1 1
1 1 1 2
1 1 3
1 2 2
1 4
2 3
5
```

54. Print all increasing sequences of length k from first n natural numbers

Given two positive integers n and k, print all increasing sequences of length k such that the elements in every sequence are from first n natural numbers.

Examples:

```
Input: k = 2, n = 3
Output: 1 2
        1 3
        2 3

Input: k = 5, n = 5
Output: 1 2 3 4 5

Input: k = 3, n = 5
Output: 1 2 3
        1 2 4
        1 2 5
        1 3 4
        1 3 5
        1 4 5
        2 3 4
        2 3 5
        2 4 5
        3 4 5
```

55. Generate all possible sorted arrays from alternate elements of two given sorted arrays

Given two sorted arrays A and B, generate all possible arrays such that first element is taken from A then from B then from A and so on in increasing order till the arrays exhausted. The generated arrays should end with an element from B.

Examples:

```
A = {10, 15, 25}
B = {1, 5, 20, 30}

The resulting arrays are:
  10 20
  10 20 25 30
  10 30
  15 20
  15 20 25 30
  15 30
  25 30
        1 3 5
        1 4 5
```

```
2 3 4
2 3 5
2 4 5
3 4 5
```

56. Merge Sort

Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is a key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

Examples:

Input: arr[] = { 12, 11, 13, 5, 6, 7 }
Output: 5 6 7 11 12 13

57. Program to find the minimum (or maximum) element of an array

Given an array, write functions to find the minimum and maximum elements in it.

Examples:

Input: arr[] = { 12, 1234, 45, 67, 1 }

Output:
Minimum element of array: 1
Maximum element of array: 1234

58. Count Inversions in an array (Using Merge Sort)

Inversion Count for an array indicates – how far (or close) the array is from being sorted. If the array is already sorted, then the inversion count is 0, but if the array is sorted in the reverse order, the inversion count is the maximum.
Formally speaking, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j

Examples:

Input: arr[] = {8, 4, 2, 1}
Output: 6

Explanation: Given array has six inversions:
(8, 4), (4, 2), (8, 2), (8, 1), (4, 1), (2, 1).

Input: arr[] = {3, 1, 2}
Output: 2

Explanation: Given array has two inversions:
(3, 1), (3, 2)

59. Combinations in a String of Digits

Given an input string of numbers, find all combinations of numbers that can be formed using digits in the same order.

Examples:

```
Input : 123
Output :1 2 3

    1 23

    12 3

    123


Input : 1234

Output : 1 2 3 4

    1 2 34

    1 23 4

    1 234

    12 3 4

    12 34

    123 4

    1234
```

60. Sum triangle from array

Given an array of integers, print a sum triangle from it such that the first level has all array elements. From then, at each level number of elements is one less than the previous level and elements at the level is be the Sum of consecutive two elements in the previous level.

Examples:

```
Input : A = {1, 2, 3, 4, 5}
Output : [48]
      [20, 28]
      [8, 12, 16]
      [3, 5, 7, 9]
      [1, 2, 3, 4, 5]

Explanation :
Here,   [48]
      [20, 28] -->(20 + 28 = 48)
      [8, 12, 16] -->(8 + 12 = 20, 12 + 16 = 28)
      [3, 5, 7, 9] -->(3 + 5 = 8, 5 + 7 = 12, 7 + 9 = 16)
```

[1, 2, 3, 4, 5] -->(1 + 2 = 3, 2 + 3 = 5, 3 + 4 = 7, 4 + 5 = 9)

61. QuickSort

Like Merge Sort, QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

1. Always pick first element as pivot.
2. Always pick last element as pivot (implemented below)
3. Pick a random element as pivot.
4. Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

Examples:

Input: arr[] = { 12, 11, 13, 5, 6, 7 }
Output: 5 6 7 11 12 13

62. Permutations - Words - 1

1. You are given a word (may have one character repeat more than once).
2. You are required to generate and print all arrangements of these characters.

Examples:

Input: aabb

Output:

```
aabb
abab
abba
baab
baba
bbaa
```

63. Words - K Selection - 1

1. You are given a word (may have one character repeat more than once).
2. You are given an integer k.

2. You are required to generate and print all ways you can select k distinct characters out of the word.

Examples:

Input:
aabbbccdde
3

Output:

abc
abd
abe
acd
ace
ade
bcd
bce
bde
cde

64. Words - K Selection - 1

1. You are given a word (may have one character repeat more than once).
2. You are given an integer k.
3. You are required to generate and print all k length words (of distinct chars) by using chars of the
    word.

Examples:

Input:
aabbbccdde
3

Output:

abc abd abe acb adb aeb acd ace adc aec ade aed bac bad bae cab dab eab cad cae dac eac dae ead bca bda bea cba dba eba cda cea dca eca dea eda bcd bce bdc bec bde bed cbd cbe dbc ebc dbe ebd cdb ceb dcb ecb deb edb cde ced dce ecd dec edc

65. Coin Change

Given a value N, if we want to make change for N cents, and we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins, how many ways can we make the change? The order of coins doesn't matter.
For example, for N = 4 and S = {1,2,3}, there are four solutions: {1,1,1,1},{1,1,2},{2,2},{1,3}. So output should be 4. For N = 10 and S = {2, 5, 3, 6}, there are five solutions: {2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}. So the output should be 5.

Examples:

Input:
arr[] = { 1, 2, 3 };

Output:
4

66. N Queen Problem

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. For example, following is a solution for 4 Queen problem.

Examples:
Input:
```
board[N][N] = { { 0, 0, 0, 0 },
              { 0, 0, 0, 0 },
              { 0, 0, 0, 0 },
              { 0, 0, 0, 0 } };
```

Output:

```
0  0  1  0
1  0  0  0

0  0  0  1

0  1  0  0
```

67. Maximum score possible after performing given operations on an Array

Given an array A of size N, the task is to find the maximum score possible of this array. The score of an array is calculated by performing the following operations on the array N times:

If the operation is odd-numbered, the score is incremented by the sum of all elements of the current array.

If the operation is even-numbered, the score is decremented by the sum of all elements of the current array.

After every operation, either remove the first or the last element of the remaining array.

Examples:

Input: A = {1, 2, 3, 4, 2, 6}
Output: 13
Explanation:
The optimal operations are:
1. Operation 1 is odd.
-> So add the sum of the array to the score: Score = 0 + 18 = 18

-> remove 6 from last,
-> new array A = [1, 2, 3, 4, 2]
2. Operation 2 is even.
-> So subtract the sum of the array from the score: Score = 18 – 12 = 6
-> remove 2 from last,
-> new array A = [1, 2, 3, 4]
3. Operation 1 is odd.
-> So add the sum of the array to the score: Score = 6 + 10 = 16
-> remove 4 from last,
-> new array A = [1, 2, 3]
4. Operation 4 is even.
-> So subtract the sum of the array from the score: Score = 16 – 6 = 10
-> remove 1 from start,
-> new array A = [2, 3]
5. Operation 5 is odd.
-> So add the sum of the array to the score: Score = 10 + 5 = 15
-> remove 3 from last,
-> new array A = [2]
6. Operation 6 is even.
-> So subtract the sum of the array from the score: Score = 15 – 2 = 13
-> remove 2 from first,
-> new array A = []
The array is empty so no further operations are possible.
Input: A = [5, 2, 2, 8, 1, 16, 7, 9, 12, 4]
Output: 50

68. Generate all numbers up to N in Lexicographical Order

Given an integer N, the task is to print all numbers up to N in Lexicographical order.

Examples:

Input: N = 15
Output:
1 10 11 12 13 14 15 2 3 4 5 6 7 8 9

Input: N = 19
Output:
1 10 11 12 13 14 15 16 17 18 19 2 3 4 5 6 7 8 9

69. Print all the palindromic permutations of given string in alphabetic order

Given a string str of size n. The problem is to print all the palindromic permutations of str in alphabetic order if possible else print “-1”.

Examples:

Input : str = "aabb"
Output :

abba

baab

Input : malayalam

Output :
aalmymlaa
aamlylmaa
alamymala
almayamla
amalylama
amlayalma
laamymaal
lamayamal
lmaayaaml
maalylaam
malayalam
mlaayaalm

70. Partition of a set into K subsets with equal sum

Given an integer array of N elements, the task is to divide this array into K non-empty subsets such that the sum of elements in every subset is same. All elements of this array should be part of exactly one partition.

Examples:

Input : arr = [2, 1, 4, 5, 6], K = 3
Output : Yes
we can divide above array into 3 parts with equal
sum as [[2, 4], [1, 5], [6]]

Input : arr = [2, 1, 5, 5, 6], K = 3
Output : No
It is not possible to divide above array into 3
parts with equal sum

71. Tug of War

Given a set of n integers, divide the set in two subsets of n/2 sizes each such that the difference of the sum of two subsets is as minimum as possible. If n is even, then sizes of two subsets must be strictly n/2 and if n is odd, then size of one subset must be (n-1)/2 and size of other subset must be (n+1)/2.
For example, let given set be {3, 4, 5, -3, 100, 1, 89, 54, 23, 20}, the size of set is 10. Output for this set should be {4, 100, 1, 23, 20} and {3, 5, -3, 89, 54}. Both output subsets are of size 5 and sum of elements in both subsets is same (148 and 148).
Let us consider another example where n is odd. Let given set be {23, 45, -34, 12, 0, 98, -99, 4, 189, -1, 4}. The output subsets should be {45, -34, 12, 98, -1} and {23, 0, -99, 4, 189, 4}. The sums of elements in two subsets are 120 and 121 respectively.

The following solution tries every possible subset of half size. If one subset of half size is formed, the remaining elements form the other subset. We initialize current set as empty and one by one build it. There are two possibilities for every element, either it is part of current set, or it is part of the remaining elements (other subset). We consider both possibilities for every element. When the size of current set becomes n/2, we check whether this solutions is better than the best solution available so far. If it is, then we update the best solution.
Following is the implementation for Tug of War problem. It prints the required arrays.

Examples:

Input : arr = {23, 45, -34, 12, 0, 98, -99, 4, 189, -1, 4}

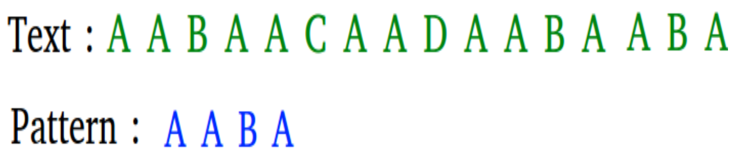
Output :
The first subset is: 45 -34 12 98 -1
The second subset is: 23 0 -99 4 189 4

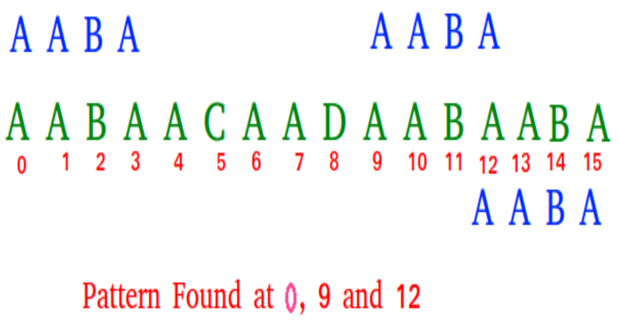
72. Pattern Searching

The Pattern Searching algorithms are sometimes also referred to as String Searching Algorithms and are considered as a part of the String algorithms. These algorithms are useful in the case of searching a string within another string.

Examples:

The Pattern Searching algorithms are sometimes also referred to as String Searching Algorithms and are considered as a part of the String algorithms. These algorithms are useful in the case of searching a string within another string.





73. Word Break Problem

Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words. See following examples for more details.
This is a famous Google interview question, also being asked by many other companies now a days.

Examples:

Consider the following dictionary
{ i, like, sam, sung, samsung, mobile, ice,
  cream, icecream, man, go, mango}

Input: ilike
Output: Yes
The string can be segmented as "i like".

Input: ilikesamsung
Output: Yes
The string can be segmented as "i like samsung"
or "i like sam sung".

74. Remove Invalid Parenthesis

An expression will be given which can contain open and close parentheses and optionally some characters, No other operator will be there in string. We need to remove minimum number of parentheses to make the input string valid. If more than one valid output are possible removing same number of parentheses then print all such output.

Examples:

Input : str = “()())()” -
Output : ()()() (())()
There are two possible solutions
"()()()" and "(())()"

Input : str = (v)())()
Output : (v)()() (v())()

75. Find Maximum number possible by doing at-most K swaps

Given a positive integer, find the maximum integer possible by doing at-most K swap operations on its digits.

Examples:

Input: M = 254, K = 1
Output: 524

Swap 5 with 2 so number becomes 524

Input: M = 254, K = 2
Output: 542
Swap 5 with 2 so number becomes 524
Swap 4 with 2 so number becomes 542

Input: M = 68543, K = 1
Output: 86543
Swap 8 with 6 so number becomes 86543

76. Write a program to print all permutations of a given string

A permutation, also called an “arrangement number” or “order,” is a rearrangement of the elements of an ordered list S into a one-to-one correspondence with S itself. A string of length n has n! permutation.

Examples:

Input: string str = "ABC";

Output:

```
ABC
ACB
BAC
BCA
CBA
CAB
```

77. Minimum number of jumps to reach end

Given an array of integers where each element represents the max number of steps that can be made forward from that element. Write a function to return the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, they cannot move through that element. If the end isn’t reachable, return -1.

Examples:

```
Input: arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}
Output: 3 (1-> 3 -> 8 -> 9)
Explanation: Jump from 1st element
to 2nd element as there is only 1 step,
now there are three options 5, 8 or 9.
If 8 or 9 is chosen then the end node 9
can be reached. So 3 jumps are made.
```

```
Input:  arr[] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
Output: 10
Explanation: In every step a jump
is needed so the count of jumps is 10.
```

78. Rat in a Maze

A Maze is given as N*N binary matrix of blocks where source block is the upper left most block i.e., maze[0][0] and destination block is lower rightmost block i.e., maze[N-1][N-1]. A rat starts from source and has to reach the destination. The rat can move only in two directions: forward and down.

In the maze matrix, 0 means the block is a dead end and 1 means the block can be used in the path from source to destination. Note that this is a simple version of the typical Maze problem. For example, a more complex version can be that the rat can move in 4 directions and a more complex version can be with a limited number of moves.

Examples:

```
Input:
maze[N][N] = { { 1, 0, 0, 0 },
             { 1, 1, 0, 1 },
             { 0, 1, 0, 0 },
             { 1, 1, 1, 1 } };
```

Output:

```
 1  0  0  0
 1  1  0  0
 0  1  0  0
 0  1  1  1
```

79. Count number of ways to reach destination in a Maze

Given a maze with obstacles, count number of paths to reach rightmost-bottommost cell from topmost-leftmost cell. A cell in given maze has value -1 if it is a blockage or dead end, else 0.
From a given cell, we are allowed to move to cells (i+1, j) and (i, j+1) only.

Examples:

```
Input: maze[R][C] =  {{0,  0, 0, 0},
                      {0, -1, 0, 0},
                      {-1, 0, 0, 0},
                      {0,  0, 0, 0}};
Output: 4
There are four possible paths as shown in
below diagram.
```

80. Gold Mine Problem

Given a gold mine of n*m dimensions. Each field in this mine contains a positive integer which is the amount of gold in tons. Initially the miner is at first column but can be at any row. He can move only (right->,right up /,right down\) that is from a given cell, the miner can move to the cell diagonally up towards the right or right or diagonally down towards the right. Find out maximum amount of gold he can collect.

Examples:

```
Input : mat[][] = {{1, 3, 3},
                   {2, 1, 4},
                  {0, 6, 4}};
Output : 12
{(1,0)->(2,1)->(2,2)}

Input: mat[][] = { {1, 3, 1, 5},
                   {2, 2, 4, 1},
                   {5, 0, 2, 3},
                   {0, 6, 1, 2}};
Output : 16
(2,0) -> (1,1) -> (1,2) -> (0,3) OR
(2,0) -> (3,1) -> (2,2) -> (2,3)

Input : mat[][] = {{10, 33, 13, 15},
                  {22, 21, 04, 1},
                  {5, 0, 2, 3},
                  {0, 6, 14, 2}};
Output : 83
```