Task 11: Recursion and its concepts C03-K3

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
Given n, calculate F(n).
Example 1:
Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.
Example 2:
Input: n = 3
Output: 2
Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.
Example 3:
Input: n = 4
Output: 3
Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3.
Constraints:
0 <= n <= 30

Algorithm:

1. If n is 0, return 0
2. If n is 1, return 1
3. Initialize variables a and b to 0 and 1, respectively
4. For i from 2 to n, calculate the next Fibonacci number by setting a = b and b = a + b
5. Return b as the nth Fibonacci number

Program:

```
#include <stdio.h>
int fibonacci(int n) {
   if (n <= 1) {
      return n;
   }
  return fibonacci(n-1) + fibonacci(n-2);
}
int main() {
   int n = 10; // example number

 printf("The %dth Fibonacci number is %d\n", n, fibonacci(n));
   return 0;
}
```

OUTPUT:

```
The 10th Fibonacci number is 55
```

Problem: The Tribonacci sequence Tn is defined as follows:
T0 = 0, T1 = 1, T2 = 1, and Tn+3 = Tn + Tn+1 + Tn+2 for n >= 0.
Given n, return the value of Tn.
Example 1:
Input: n = 4
Output: 4

Explanation
:
T_3 = 0 + 1 + 1 = 2
T_4 = 1 + 1 + 2 = 4
Example 2:
Input: n = 25
Output: 1389537
Constraints:
0 <= n <= 37
The answer is guaranteed to fit within a 32-bit integer, ie. answer <= 2^31 - 1.

Álgorithm:

```
1. If n is 0, return 0.
2. If n is 1 or 2, return 1.
3. Initialize variables t0 = 0, t1 = 1, t2 = 1, and t3 = 2.
4. Loop from 3 to n, and at each iteration, calculate the value of the next term in the sequence by
   setting t3 = t0 + t1 + t2, then update the values of t0, t1, and t2 as follows: t0 = t1, t1 = t2, t2 =
   t3.
5. Return the value of t3.
```

Program:

```
#include <stdio.h>
int fibonacci(int n) {
   if (n <= 1) {
      return n;
   }
   return fibonacci(n-1) + fibonacci(n-2);
}
int main() {
   int n = 10; // example number
   printf("The %dth Fibonacci number is %d\n", n, fibonacci(n));
   return 0;
}
```

OUTPUT:

```
Enter the number 0f elements:15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Result: Thus the Program is executed and verified successfully.

Task 12: Elementary data structure algorithms-based Questions and finding complexity.-CO4-K3

Little Shino and Paris

Given a permutation of number from 1 to N. Among all the subarrays, find the number of unique pairs such that and a is maximum and b is second maximum in that subarray.

Input:

First line contains an integer, N . Second line contains N space separated distinct integers, , denoting the permutation.

Output:

Print the required answer.

SAMPLE INPUT

5

1 2 3 4 5

SAMPLE

OUTPUT 4

Q UEUE

Algorithm:

1. Read the input values of N and the permutation array A.
2. Initialize a variable count to 0, to keep track of the number of valid pairs.
3. Loop over all possible subarrays of A, with nested loops i and j:
    a. Find the maximum element maxval and its index maxidx in the subarray [i,j].
    b. Find the second maximum element secondmax in the subarray [i,j] using a separate loop or a priority queue.
    c. If maxval is unique in the subarray and secondmax is also present, increment count.
4. Print the final value of count.

Program:

```
#include
<stdio.h> int
main() {
  int n, i, j, max, second_max, count = 0;
  scanf("%d", &n);
  int arr[n];
  for (i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }

  for (i = 0; i < n; i++) {
    max = arr[i];
    second_max = -1;
    for (j = i + 1; j < n; j++) {
      if (arr[j] > max) {
        second_max =
        max; max = arr[j];
      }
      else if (arr[j] > second_max) {
```

```
                second_max = arr[j];
            }
            if (second_max != -1 && arr[i] == second_max) {
                count++;
                break;
            }
        }

    }
    printf("%d", count);
    return 0;
}
```

OUTPUT:

```
SAMPLE
INPUT 5
1 2 3 4 5
SAMPLE
OUTPUT 4
```

FUN SORT

Divide the given array of size equal to the sum of first n natural numbers into subarrays of size 1,2,3,,n. For the xth subarray of size x – if x is odd then sort it in descending order, otherwisesort it in ascending order. Output the sum of 1st element of each subarray.

Algorithm:

1. Calculate the sum of first n natural numbers:
    - Initialize a variable sum to 0
    - Loop through the range from 1 to n+1
    - Add each number to the sum variable
2. Create an array of size equal to the sum calculated in step 1
3. Loop through the range from 1 to n+1:
    - If x is odd:
        - Get the subarray of size x starting from the index sum-x
        - Sort the subarray in descending order
    - Else:
        - Get the subarray of size x starting from the index sum-x
        - Sort the subarray in ascending order
    - Add the first element of the subarray to a variable called result
    - Increase the sum of the indices of the subarray to x
4. Return the variable result as the output

PROGRAM:

```
#include <stdio.h>

#include <stdlib.h>
int main() {
    int T, n, i, j, k, sum, size;
    scanf("%d", &T);
    while (T--) {
        scanf("%d", &n);
        size = (n * (n + 1)) / 2;
        int *arr = (int*) malloc(sizeof(int) * size);
        for (i = 0; i < size; i++) {
            scanf("%d", &arr[i]);
        }
        sum = 0;
        k = 0;
        for (i = 1; i <= n; i++) {
            if (i % 2 != 0) {

                for (j = 0; j < i; j++) {
                    int max = j;
                    for (int l = j + 1; l < i; l++) {
                        if (arr[l + k] > arr[max + k]) {
                            max = l;
                        }
                    }
                    int temp = arr[j + k];
                    arr[j + k] = arr[max +
                    k]; arr[max + k] = temp;
                }
                sum +=
                arr[k]; k++;
            }
            else {
                for (j = 0; j < i; j++) {
                    int min = j;
                    for (int l = j + 1; l < i; l++) {
                        if (arr[l + k] < arr[min + k]) {
                            min = l;
                        }
                    }
                    int temp = arr[j + k];
                    arr[j + k] = arr[min + k];
                    arr[min + k] = temp;
                }
                sum +=
                arr[k]; k += i;
            }
        }
        printf("%d\n",
        sum); free(arr);
    }
    return 0;
}
```
OUTPUT:

Input:

1

3
1 2 3 4 5 6
Output:
9

Result: Thus the program is executed and verified successfully.

Task 13 : Partial Code Completion.-CO5-K3

1. How many times CppBuzz.com is printed?

```
int main()
{
int a = 0;
while(a++)
;
{
   printf("CppBuzz.com");
}
return 0;
}
```

OUTPUT: infinite “CppBuzz.com”

2. What is output of below program?

```
int main()
{
int i,j,count;
count=0;
for(i=0; i<5; i++);
{
        for(j=0;j<5;j++);
        {
           count++;
        }
}
printf("%d",count);
return 0;
}
```

OUTPUT: 25

3. What is output of below program?

```
int main()
{
 int i;
 for(i=0; i<5; i++);
 {
   printf("cppbuzz");
 }
 return 0;
}
```

OUTPUT:

```
Cppbuzz
```

```
Cppbuz
z
Cppbuz
z
cppbuzz
cppbuzz
```

4. What is output of below program?

```
int main()
{
int i,j,k,count;
count=0;
for(i=0;i<5;i++
)
{
   for(j=0;j<5;j++)
   {
      count++;
   }
}
printf("%d",count);
return 0;
}
```

OUTPUT: 25

5. What is output of below program?

```
int main()
{
 int i,j;
 for(i = 0,j=0;i<5;i++)
 {
   printf("%d%d--",i,j);
 }
 return 0;
}
```

OUTPUT:

```
00--10--20--30--40--
```

6. What is output of below program?

```
int main()
{
 int i;
 for(i=0; i<5; ++i++)
 {
   printf("Hello");
 }
 return 0;
}
```

ERROR: invalid increment operand

7. What is the output of below program?

```
int main()
```

```
{
 int i;
 for(i = 0,i<5,i++)
 {
   printf("Hello");
 }
 return 0;
}
```

OUTPUT:

```
Hell
o
Hell
o
Hell
o
Hell
o
Hell
o
```

8. What is output of below program?

```
int main()
{
 for(; ;);
 for(; ;);
   printf("Hello")
; return 0;
}
```

NO OUTPUT

9. What is the output of below program?

```
int main()

{
for(; ;)
for(; ;)
   printf("Hello..");

return 0;
}
```

OUTPUT:
INFINITE “Hello..”

10. What is output of below code?

```
int main()
{
char name[]="Cppbuz";
function “size_of()”
int len;
int
size;
len = strlen(name);
size =
size_of(name);
printf("%d,%d",len,size);
return 0;
}
```

ERROR: invalid/undeclared

11. What is output of below program?

```
int main()
{
const int a = 10;                              ERROR: cannot increment a constant
variable
printf("%d",++a);
return 0;
}
```
12. What is output of below program?

```
int main()
{
const int a = 10;
printf("%d",++a + ++a);                                   ERROR: cannot increment a constant
variable
return 0;
}
```

13. What is output of below program?

```
int main()
{
const int a = 10;
printf("%d",--a);
return 0;                                                ERROR: decrement of read-only variable 'a'
}
```

14. What is storage class for variable A in below code?

```
int main()
{
int A;                                                   RESULT: Class of A  integer
A = 10;
printf("%d", A);
return 0;
}
```

15. #include
"stdio.h" char * f();

```
char a = 'a';                                            RESULT:     Class of a character
int main(int argc, char *argv[])
{
char *temp = f();
printf("%&", temp);
return 0;
}
char *f()
{ return &a;
}
```

Task 14:Practice Problems for Coding Preparation-CO5-K3

N QUEENS

Given a chess board having N×N cells, you need to place N queens on the board in such a way that no queen attacks any other queen.

Input:

The only line of input consists of a single integer denoting N.

Output:

If it is possible to place all the N queens in such a way that no queen attacks another queen, then print N lines having N integers. The integer in ith line and jth column will denote the cell (i,j) of the board and should be 1 if a queen is placed at (i,j) otherwise 0. If there are more than way of placing queens print any of them. If it is not possible to place all N queens in the desired way, then print "Not possible" (without quotes).

Constraints

: 1≤N≤10.

SAMPLE INPUT

```
 4
```

SAMPLE OUTPUT

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

Algorithm:

1. Define a function to check if a queen can be placed in a given cell of the board.
2. Define a function to recursively place queens on the board.
3. In the recursive function, check if all the queens have been placed. If yes, return the board configuration.
4. If not, loop through all the cells in the current row and check if a queen can be placed in that cell using the isSafe() function.
5. If a queen can be placed in a cell, mark the cell as occupied and recursively call the function for the next row.
6. If the recursive call returns a board configuration, return it.
7. If the recursive call does not return a board configuration, unmark the cell and continue the loop for the next cell.
8. If no queen can be placed in any cell of the current row, return None.
9. In the main function, call the recursive function to place the queens and print the board configuration.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_N 10
```

```
int n;
int board[MAX_N][MAX_N];

int is_valid(int row, int col) {
    int i, j;
    for (i = 0; i < n; i++) {

        if (board[row][i] || board[i][col])
            return 0;
    }
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--) {
        if (board[i][j])
            return 0;
    }
    for (i = row, j = col; i < n && j >= 0; i++, j--) {
        if (board[i][j])
            return 0;
    }
    return 1;
}

int solve(int col) {
    int row, i;
    if (col >=
        n) return
        1;
    for (row = 0; row < n; row++) {
        if (is_valid(row, col)) {
            board[row][col] =
            1; if (solve(col + 1))
                return 1;
            board[row][col] = 0;
        }
    }
    return 0;
}
int main()
    { int i, j;
    scanf("%d", &n);
    if (solve(0)) {
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++)
            {
                printf("%d ", board[i][j]);
            }
            printf("\n");
        }
    } else {
        printf("Not possible\n");
    }
    return 0;
}
```

OUTPUT:

```
SAMPLE INPUT
 4
SAMPLE OUTPUT
 0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
```

THE CASTLE GATE

Gudi, a fun loving girl from the city of Dun, travels to Azkahar - a strange land beyond the mountains. She arrives at the gates of Castle Grey, owned by Puchi,the lord of Azkahar to claim the treasure that it guards. However, destiny has other plans for her as she has to move through floors, crossing obstacles on her way to reach the treasure.

The gates of the castle are closed. An integer N is engraved on the gates. A writing on the wall says Tap the gates as many times as there are unordered pairs of distinct integers from 1 to N whose bit- wise XOR does not exceed N.

Help her find the number of the times she has to tap.

Input:

First line contains an integer T, T testcases follow.

Each testcase consists of an integer N.

Output:

Print the answer to each testcase in a newline. Constraints:

```
SAMPLE
INPUT 3
4
6
8
SAMPLE
OUTPUT 3
12
```

Algorithm:

```
1. Read the input value of T.
2. Repeat the following steps for T test cases:
    a. Read the value of N.
    b. Initialize the variable count to 0.
    c. Iterate i from 1 to N-1.
        1. Iterate j from i+1 to N.
    d. If (i XOR j) is less than or equal to N, increment count.
    e. Print count.
3. End
```

Program:

```
#include
```

```
<stdio.h>
int countSetBits(int n) {
    int count = 0;
    while (n > 0) {
        n &= (n - 1);
        count++;
    }
    return count;
}
int main()
    { int t, n;
    scanf("%d", &t);
    while (t--) {
        scanf("%d", &n);

        int count = 0;
        for (int i = 1; i <= n; i++) {
            for (int j = i + 1; j <= n; j++) {
                if ((i ^ j) <= n) {
                    count++;
                }
            }
        }
        printf("%d\n", count);
    }
    return 0;
}
```

OUTPUT:

Sample
input: 3 4 6 8

Sample output:

```
3
12
21
```

Result: Thus the Program is executed and verified successfully.