Programming Test 1

1. You can use any OOPS language to code the solution
2. Evaluation of your code will be based on patterns used , maintainability , scalability , performance ,loose coupling of your solution.
3. Write a Java program to calculate Fibonacci Series up to n numbers. Or
4. Write a Java Program to check whether the given array is Mirror Inverse or not.
5. Write a Java program to implement HeapSort Algorithm. Or
6. Write a Java Program to find the Transpose of a given Matrix.
7. Given 3 numbers {1, 3, 5}, The task is to tell the total number of ways we can form a number **N** using the sum of the given three numbers.
8. Given a linked list consisting of L nodes and given a number N. The task is to find the Nth node from the end of the linked list.

**Coding Page**

**Q.1** **Write a Java program to calculate Fibonacci Series up to n numbers.**

**Solution:**

**public** **class** problem1 {

**static** **int** *n1*=0,*n2*=1,*n3*=0;

**static** **void** printFibonacci(**int** count){

**if**(count>0){

*n3* = *n1* + *n2*;

*n1* = *n2*;

*n2* = *n3*;

System.***out***.print(" "+*n3*);

*printFibonacci*(count-1);

}

}

**public** **static** **void** main(String args[]){

**int** count=10;

System.***out***.print(*n1*+" "+*n2*);//printing 0 and 1

*printFibonacci*(count-2);//n-2 because 2 numbers are already printed

}

}

**Q.4** **Write a Java Program to find the Transpose of a given Matrix.**

**Solution:**

**public** **class** Transpose {

**public** **static** **void** main(String[] args) {

**int** row = 2, column = 3;

**int**[][] matrix = { {2, 3, 4}, {5, 6, 4} };

// Display current matrix

*display*(matrix);

// Transpose the matrix

**int**[][] transpose = **new** **int**[column][row];

**for**(**int** i = 0; i < row; i++) {

**for** (**int** j = 0; j < column; j++) {

transpose[j][i] = matrix[i][j];

}

}

// Display transposed matrix

*display*(transpose);

}

**public** **static** **void** display(**int**[][] matrix) {

System.***out***.println("The matrix is: ");

**for**(**int**[] row : matrix) {

**for** (**int** column : row) {

System.***out***.print(column + " ");

}

System.***out***.println();

}

}

}

**Q.5 Given 3 numbers {1, 3, 5}, The task is to tell the total number of ways we can form a number N using the sum of the given three numbers.**

**Solution:**

**public** **class** Etraveli {

// Function to count the

// number of ways to represent

// n as sum of 1, 3 and 5

**static** **int** countWays(**int** n)

{

**int** DP[] = **new** **int**[n + 1];

// base cases

DP[0] = DP[1] = DP[2] = 1;

DP[3] = 2;

// iterate for all values from 5 to n

**for** (**int** i = 5; i <= n; i++)

DP[i] = DP[i - 1] + DP[i - 3]

+ DP[i - 5];

**return** DP[n];

}

// driver code

**public** **static** **void** main(String[] args)

{

**int** n = 10;

System.***out***.println(*countWays*(n));

}

}

**Q.6 Given a linked list consisting of L nodes and given a number N. The task is to find the Nth node from the end of the linked list.**

**Solution:**

#include <stdio.h>

#include <stdlib.h>

/\* Link list node \*/

typedef struct Node {

int data;

struct Node\* next;

} Node;

/\* Function to get the nth node from the last of a linked

\* list\*/

void printNthFromLast(Node\* head, int N)

{

int len = 0, i;

Node\* temp = head;

// Count the number of nodes in Linked List

while (temp != NULL) {

temp = temp->next;

len++;

}

// Check if value of N is not

// more than length of the linked list

if (len < N)

return;

temp = head;

// Get the (len-N+1)th node from the beginning

for (i = 1; i < len - N + 1; i++)

temp = temp->next;

printf("%d", temp->data);

return;

}

void push(struct Node\*\* head\_ref, int new\_data)

{

/\* Allocate node \*/

Node\* new\_node = (Node\*)malloc(sizeof(Node));

/\* Put in the data \*/

new\_node->data = new\_data;

/\* link the old list of the new node \*/

new\_node->next = (\*head\_ref);

/\* move the head to point to the new node \*/

(\*head\_ref) = new\_node;

}

// Driver's Code

int main()

{

/\* Start with the empty list \*/

struct Node\* head = NULL;

// create linked 35->15->4->20

push(&head, 20);

push(&head, 4);

push(&head, 15);

push(&head, 35);

// Function call

printNthFromLast(head, 4);

return 0;

}