**ROLL NO 2022-CS-111**

**MUHAMMAD FAIZAN ASIM**

**int SumNatural(int n)**

**{**

**if(n == 0)**

**{**

**return 0;**

**}**

**return n + SumNatural(n - 1);**

**}**

**// 2**

**int SumOdd(int n)**

**{**

**if(n == 1)**

**{**

**return 1;**

**}**

**return n + SumOdd(n - 2);**

**}**

**// 3**

**int SumTwoPower(int n)**

**{**

**if(n == 0)**

**{**

**return 1;**

**}**

**return pow(2,n) + SumTwoPower(n - 1);**

**}**

**// 4**

**int SumThreePower(int n)**

**{**

**if(n == 0)**

**{**

**return 1;**

**}**

**return pow(3,n) + SumThreePower(n - 1);**

**}**

**// 5**

**int SumThreeDivide(int n)**

**{**

**if(n == 0)**

**{**

**return 1;**

**}**

**return n + SumThreeDivide(n/3);**

**}**

**// 6**

**int SumTwoDivide(int n)**

**{**

**if(n == 0)**

**{**

**return 1;**

**}**

**return n + SumTwoDivide(n/2);**

**}**

**QUESTION 2**

**string DecimalToBinary(int n)**

**{**

**if(n/2 == 0)**

**{**

**return "1";**

**}**

**return DecimalToBinary(n/2) + to\_string((n%2));**

**}**

**int DecimalToBinaryInter(int n)**

**{**

**if(n/2 == 0)**

**{**

**return 1;**

**}**

**return DecimalToBinaryInter(n/2)\*10 + ((n%2));**

**}**

**int GCD(int A,int B)**

**{**

**if(A%B == 0)**

**{**

**return B;**

**}**

**return (B,A%B);**

**}**

**QUESTION 3**

**int searchFirstEntitty(int arr[],int size,int ind,int find)**

**{**

**if(ind == size)**

**{**

**return -1;**

**}**

**if(arr[ind] == find)**

**{**

**return ind;**

**}**

**return searchFirstEntitty(arr,size,ind+1,find);**

**}**

**int searchLastEntitty(int arr[],int size,int ind,int find)**

**{**

**if(ind == 0)**

**{**

**return -1;**

**}**

**if(arr[ind] == find)**

**{**

**return ind;**

**}**

**return searchLastEntitty(arr,size,ind-1,find);**

**}**

**int BinaryEntitty(int arr[],int size,int ind,int find)**

**{**

**int left = ind;**

**int right = size;**

**if(left <= right)**

**{**

**int mid = left + (right-left)/2;**

**if(arr[mid] == find)**

**{**

**return mid;**

**}**

**else if(arr[mid] > find)**

**{**

**return BinaryEntitty(arr,mid - 1,ind,find);**

**}**

**else**

**{**

**return BinaryEntitty(arr,size,mid + 1,find);**

**}**

**}**

**return -1;**

**}**

**QUESTION 4**

**int Power(int x,int y, int M)**

**{**

**if(y == 0)**

**{**

**return 1%M;**

**}**

**return (x \* Power(x,y-1,M))% M;**

**}**

**int PowerByLog(int x,int y,int M)**

**{**

**int res = 0;**

**if(y == 0)**

**{**

**return 1%M;**

**}**

**if(y%2 == 0)**

**{**

**int hlf = PowerByLog(x,y/2,M);**

**res = (hlf\*hlf)%M;**

**}**

**else{**

**int hlf = PowerByLog(x,(y-1)/2,M);**

**res = ((hlf\*hlf)%M \* (x % M )) % M;**

**}**

**return (res+M)%M;**

**}**

**QUESTION 5**

**int Add(int x, int y)**

**{**

**if(y ==0 )**

**{**

**return 0;**

**}**

**return x + Add(x,y-1);**

**}**

**int MUL(int A,int B)**

**{**

**return Add(A,B);**

**}**

**int MULTLOG2Y(int A, int B) {**

**if (B == 0) {**

**return 0;**

**} else if (B % 2 == 0) {**

**int half\_product = MULTLOG2Y(A, B /2 );**

**return Add(half\_product, half\_product);**

**} else {**

**int half\_product = MULTLOG2Y(A, (B - 1)/2);**

**return Add(half\_product, Add(half\_product, A));**

**}**

**}**

**QUESTION 6**

**void divMod(int x,int y,int &ans,int &rem)**

**{**

**if(rem < y)**

**{**

**return;**

**}**

**rem = rem - y;**

**ans++;**

**return divMod(x,y,ans,rem);**

**}**

**void divModLog(int A, int B, int &qu, int &rem) {**

**if (A < B) {**

**qu = 0;**

**rem = A;**

**} else {**

**int multiple = 1;**

**int tempB = B;**

**while (A >= (tempB << 1)) {**

**tempB <<= 1;**

**multiple <<= 1;**

**}**

**divModLog(A - tempB, B, qu, rem);**

**qu += multiple;**

**}**

**}**

**QUESTION 7**

**int thirdFib(int ind) {**

**if (ind <= 1) {**

**return ind;**

**}**

**vector<int> temp(ind + 1);**

**temp[0] = 0;**

**temp[1] = 1;**

**for (int i = 2; i <= ind; ++i) {**

**temp[i] = temp[i - 1] + temp[i - 2];**

**}**

**for(int i=0;i<=ind;i++)**

**{**

**cout<<temp[i]<<" ";**

**}**

**}**

**QUESTION 8**

**vector<int> tempVector;**

**//3**

**int sumMem(int n) {**

**switch (n) {**

**case 0:**

**return 0;**

**case 1:**

**return 1;**

**case 2:**

**return 2;**

**case 3:**

**return 3;**

**default:**

**if (tempVector[n] != 0) {**

**return tempVector[n];**

**}**

**tempVector[n] = sumMem(n - 1) + sumMem(n - 2) +sumMem(n - 3);**

**return tempVector[n];**

**}**

**}**

**//1**

**int sum1(int index) {**

**if (index == 1) {**

**return 1;**

**}**

**if (index == 2) {**

**return 2;**

**}**

**if (index == 3) {**

**return 3;**

**}**

**return sum1(index - 1) + sum1(index - 2) + sum1(index - 3);**

**}**

**//2**

**int sumBottomUp(int index) {**

**if (index <= 0) {**

**return 0;**

**}**

**if (index == 1) {**

**return 1;**

**}**

**if (index == 2) {**

**return 2;**

**}**

**if (index == 3) {**

**return 3;**

**}**

**vector<int> temp(index + 1);**

**temp[1] = 1;**

**temp[2] = 2;**

**temp[3] = 3;**

**for (int i = 4; i <= index; i++) {**

**temp[i] = temp[i - 1] + temp[i - 2] + temp[i - 3];**

**}**

**return temp[index];**

**}**

**QUESTION 9**

**void vectorPrinter(vector<int> &d)**

**{**

**for(int n : d)**

**{**

**cout<<n<<",";**

**}**

**if(d.size() == 0)**

**{**

**cout<<"0";**

**}**

**else**

**{**

**cout<<endl;**

**}**

**}**

**int Subarray(vector<int> &temp,int start,int end,vector<int> &ans)**

**{**

**if(start > end)**

**{**

**vectorPrinter(ans);**

**return 0;**

**}**

**ans.push\_back(temp[start]);**

**Subarray(temp,start+1,end,ans);**

**ans.pop\_back();**

**Subarray(temp,start+1,end,ans);**

**}**

**QUESTION 10**

**void sort(stack<int> &org)**

**{**

**if(!org.empty())**

**{**

**int top = org.top();**

**org.pop();**

**sort(org);**

**insert(org,top);**

**}**

**}**

**void insert(stack<int> &org,int top)**

**{**

**if(org.empty() || org.top() < top)**

**{**

**org.push(top);**

**return;**

**}**

**int topEle = org.top();**

**org.pop();**

**insert(org,top);**

**org.push(topEle);**

**}**

**void show(stack<int> sta) {**

**stack<int> temp =sta;**

**for(int i= 0 ; i < 5 ; i++)**

**{**

**cout << temp.top() << " ";**

**temp.pop();**

**}**

**cout << endl;**

**}**