**Algo - Class Activity - 2**

**Question # 1**

What is the following Recursive **MasterFunction** doing..... Debug on input 31, 4, 15, 28, 7

|  |
| --- |
| **string MasterFunction(int N)**  **{**  **if(N==0)**  **return "0";**  **if(N==1)**  **return "1";**  **else if(N%2==0)**  **return MasterFunction(N/2) + "0";**  **else // N%2 == 1**  **return MasterFunction(N/2) + "1";**  **}** |

**Ans : The Master Function Takes a input as a inteager and returns a string of binaray as output!**

**It checks if the “N” is even then it Divide the inteager into half and return “0” otherwise it check if it is Odd then it returns 1 and again devide the inteager .! It also check if the number is “0” or “1” and returns accordingly**

**Question # 2**

**Write a recursive function which uses divide and conquer strategy (recursive strategy - in which problem size is decreasing), Find Minimum Value/minValueIndex within the array.**

**int FindMinimum(int A[ ], int Size)** // You may call any helper function.... but it has to be recursive....

int MinHelper(vector<int>&v,int s, int e,int &min)

{

if(s >=e)

{

return min;

}

if(min> v[s])

min = v[s];

MinHelper(v, s+1,e, min);

}

int findMin(vector<int>&v)

{

int mid = (0 + v.size())/ 2;

int r1= MinHelper(v, 0, mid, v[0]);

int r2= MinHelper(v,mid+1, v.size(),v[mid+1]);

return (r1<r2 ? r1:r2);

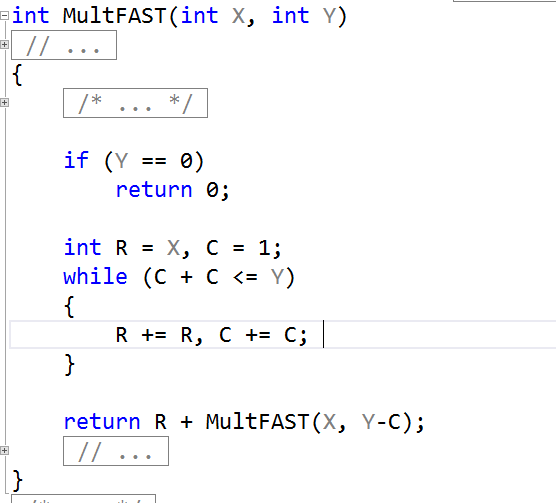
}

**Question # 3**

What is the time complexity of the following code snippet

|  |  |
| --- | --- |
| **i) What is the time complexity of f1**  **void Run(int T)**  **{**  **int K=0;**  **for(int i=0; i<=T; i++) O(N)**  **k++;**  **}**  **int f1(int N)**  **{**  **for(int i=1; i<=N; i\*=2) O(N)**  **Run(i);**  **}**  **total = O(N) + O(N) = 2N** | **ii) What is the time complexity of f2.**  **void Run2(int t)**  **{**  **int K=0; int T = pow(2, t); T^2**  **for(int i=0; i<=T; i++)**  **k++;**  **}**  **int f2(int N)**  **{**  **for(int i=1; i<=logN; i++) log N**  **Run2(i);**  **}**  **total = O(T^2) + LogN** |

**Question # 4**



**Dry run FASTMULT on FASTMULT (2, 2047) and compute the worst case running time for FASTMULT (X, Y).   
HINT - Its not log Y. Look at the dry run of FASTMULT (2, 2047).**

**(BONUS)\***

**- How can you use memoization techniques to make it better?**