|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ROLL NO: | | | | |
| KARPAGAM COLLEGE OF ENGINEERING, COIMBATORE-641032 | | | | |
| B.E (CSE,ECE,EEE,EIE,ETE)/B.TECH IT | | | SEMESTER : III | |
| 17PE01/17LE01/17EE01/17NE01/17TE01/17FE01- ADVANCED DATA STRUCTURES | | | | |
| CONTINUOUS INTERNAL ASSESSMENT: I ANSWER KEY | | | | |
| DURATION : 3 HOURS | | | DATE : 17.07.2018 | |
| MAXIMUM : 100 MARKS | | | SESSION : FN | |
| Answer All Questions | | PART- A (10 x 2 = 20 Marks) | | |
| A1 | Ascertain the running time for the following code segment. Express it using Big O notation.  void function(int n)  {  int i, j, k , count =0;  for(i=n/2; i<=n; i++)  for(j=1; j<=n; j= 2 \* j)  for(k=1; k<=n; k= k \* 2)  count++;  }  ANSWER  O(N (log n)2) | | |
| A2 | Arrange the following asymptotic functions given below in the ascending order of their growth rate.  { 2n, n2, nLogn, n!, 25, n2logn, log2n, 3/N }  ANSWER  25,3/N , log2 n, n Log n, n2,n2log n, 2n, n! | | |
| A3 | Predict the output for the following recursive code. Give the contents of each recursive call.  void fun1(int n)  {   int i = 0;   if (n > 1)       fun1(n-1);   for (i = 0; i < n; i++)       printf(" \* ");  }  ANSWER  N=5  \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* | | |
| A4 | Write the recurrence for the function f( ) and solve it using substitution method to ascertain the running time. Assume that addition can be done in constant time.  def f(n):  if n == 1:  return 1  else:  return f(n-1)+f(n-1)  ANSWER | | |
| A5 | State the running time of the following looping constructs A, B, C and D. Which among them is the most efficient in terms of their running time if ‘n’ is the size of the input?  A) for (i = 0; i < n; i++);    B) for (i = 0; i < n; i += 2);    C) for (i = 1; i < n; i \*= 2);    D) for (i = n; i > -1; i /= 2);   |  | | --- | | ANSWER  Running time of Loops:   1. O(N) 2. O(N) 3. O(LOG N) 4. Infinite loop | | Efficient Loop:  C) for (i = 1; i < n; i \*= 2); | | | |
| A6 | What is the output of following program?   |  | | --- | | # include <stdio.h>  int main()  {     char str1[] = "HackerEarth";     char str2[] = {'H', 'a', 'c', 'k', 'e', 'r', 'E', 'a', 'r',’t’,’h’};     int n1 = sizeof(str1)/sizeof(str1[0]);     int n2 = sizeof(str2)/sizeof(str2[0]);     printf("n1 = %d, n2 = %d", n1, n2);     return 0;  }  ANSWER  n1=12, n2=11 | | | |
| A7 | Given a boolean 2D array, where each row is sorted. Write a clean code to find the row with the maximum number of 1s.  Example  Input matrix  0 1 1 1  0 0 1 1  1 1 1 1 // this row has maximum 1s  0 0 0 0  Output: 2  ANSWER  for(i=0;i<4;i++)  for(j=0;j<4;j++)  {  scanf("%d",&a[i][j]);  if(a[i][j]==1)  hash[i]++;  }  int max=0;  for(i=0;i<4;i++)  if(hash[i]>max)  {  max=hash[i];  k=i;  }  printf("max row is %d",k ); | | |
| A8 | Consider a staircase of size N. Write a program that prints a staircase of size N. State its time complexity.   |  |  |  |  | | --- | --- | --- | --- | |  |  |  | # | |  |  | # | # | |  | # | # | # | | # | # | # | # |   Input Format  A single integer, n, denoting the size of the staircase.  Output Format  Print a staircase of size N using # symbols and spaces.  ANSWER  int main() {  int n, i,j;  scanf("%d", &n);  for(i=0;i<n;i++)  {  for(j=0;j<n;j++)  {  if(j<(n-1-i)) printf(" ");  else printf("#");  }  printf("\n");  }  Time Complexity:O(n2) | | |
| A9 | Assess the Big O complexity for the following real life scenarios. Justify the answer in one line.   1. Carry ‘N’ items from one room to another room one at a time. 2. Locating a patient record in Doctor’s Office. 3. Store manager gives gifts to first 10 customers in a queue irrespective of total number of customers. 4. Inserting a book in its proper place in the book shelf.  |  |  | | --- | --- | | ANSWER  1) O(N)  2) O(N)/O(LOG N) | 3)O(1)  4)O(1)/O(N) | | | |
| A10 | Design an algorithm to find the second largest element in an array of N distinct integers without using sorting technique.  ANSWER  void print2largest(int arr[], int arr\_size)  {      int i, first, second;      if (arr\_size < 2)      {          printf(" Invalid Input ");          return;      }      first = second = INT\_MIN;      for (i = 0; i < arr\_size ; i ++)      {          if (arr[i] > first)          {              second = first;              first = arr[i];          }          else if (arr[i] > second && arr[i] != first)              second = arr[i];      }      if (second == INT\_MIN)          printf("There is no second largest element\n");      else          printf("The second largest element is %dn", second);  } | | |