ROLL NO:

KARPAGAM COLLEGE OF ENGINEERING, COIMBATORE-641032

B.E CSE,ECE,EEE,EIE,ETE/B.TECH IT SEMESTER : IV

15PE05 / 15FE05 – DESIGN AND ANALYSIS OF ALGORITHMS

CONTINUOUS INTERNAL ASSESSMENT: I Answer Key

DURATION : 3 HOURS DATE : 18.01.2018

MAXIMUM : 100 MARKS SESSION : FN

Answer All Questions PART- A (10 x 2 = 20 Marks)

A1 Given a set of 'N' numbers, we can get the sorted order by first building a binary search tree containing these 'N' numbers (using TREE-INSERT( ) repeatedly to insert the numbers one by one) and then printing these numbers by an inorder tree walk. Find the worst-case and best-case running times for this sorting algorithm. 02

Ans: Worst Case: $O(n^2)$; Best Case: O(n log n)

A2 Write the recurrence relation for the following algorithm and derive its running time. Assume the function divide is called as follows: divide(0,n,numbers) where 'numbers' is an array whose values are unknown and 'n' is the length of the array. Express your answer in big O notation as a function of 'n'. 02

```
Divide(int start, int end, int numbers[]) {
  if(start>=end)
     return 0;
  int index=start+ (end-start)/2;
 divide(start,index,numbers);
 divide(index,end,numbers);
 for(int i=start;i<=end;i++)
       print(numbers[i]);
}
```

Ans:

$T(n) = 2T(n/2)+n$

$= 2[2T(n/4)+n]+n$

$= 4T(n/4)+2n$

$= 2^kT(n/2^k)+kn$

Setting k=log n, $T(n)=2^{\log n}T(0)+n\log n$

Thus, T(n) = n+n log n, so we can conclude that $T(n) \in O(n \log n)$

A3 Implement an algorithm for removing duplicate elements present in the given array. Expected running time is O(n). 02

Input 1:arr[] ={12,10,9,45,12,10,45} Output 1: arr[] = {12, 10, 9, 45}

Input 2: arr[] = {1, 2, 3, 4, 5} Output 2: arr[]={1, 2, 3, 4, 5}

Input 3: arr[] = {1, 1, 1, 1, 1} Output 3: arr[]={1}

Ans:

```
void duplicate(int a[],int hash[],int n){
int i;
for(i=0;i<n;i++)
        hash[a[i]]+=1;
for(i=0;i<n;i++) {
 if(hash[a[i]]>1) {
        printf("%d",a[i]);
        hash[a[i]]=hash[a[i]]-1;
}}}
```

A4 Analyze the following algorithm. How many right-shifts does it use, i.e., how many times step 2 is executed in the worst case? 02

```
Function increment(y)
1. If y==0 then return(1) else
2. If(y mod 2)==1 then
3.     Return(2*increment(y/2))
4. Else Return(y+1)
```

Ans: T(n) = T(n/2)+1 =O(logn)
Therefore, Step 2 will be executed logn times in worst case.

A5 An array 'A' consists of 'N' elements, each of which is red, white, or blue. We seek to sort the elements so that all the reds come before all the whites, which come before all the blues. The only operation permitted on the keys are 02

- Examine(A, i) {report the color of the ith element of A}.
- Swap(A,i,j) { swap the ith element of A with the jth element}.

Find an efficient algorithm for red-white-blue sorting. Expected time complexity is linear i.e O(N).

Ans:
```
void sort012(int a[], int arr_size) {
   int lo = 0;
   int hi = arr_size - 1;
   int mid = 0;
   while (mid <= hi)    {
      switch (a[mid])       {
      case 0:
         swap(&a[lo++], &a[mid++]);
         break;
      case 1:
         mid++;
         break;
      case 2:
         swap(&a[mid], &a[hi--]);
         break;
}}}
```

A6 Predict the output of the following function. 02

```
#include<stdio.h>
int f(int *a, int n) {
  if(n <= 0)
         return 0;
  else if(*a % 2 == 0)
        return *a + f(a+1, n-1);
  else
        return *a - f(a+1, n-1);
}
int main() {
  int a[] = {12, 7, 13, 4, 11, 6};
  printf("%d", f(a, 6));
  getchar();
  return 0;
}
```

Ans:
```
 f(add(12), 6) /*Since 12 is first element. a contains address of 12 */
          |
 12 + f(add(7), 5) /* Since 7 is the next element, a+1 contains address of 7 */
          |
    7 - f(add(13), 4)
           |
        13 - f(add(4), 3)
              |
           4 + f(add(11), 2)
                 |
              11 - f(add(6), 1)
                    |
```

6 + 0

So, the final returned value is 12 + (7 – (13 – (4 + (11 – (6 + 0))))) = 15

A7 We are comparing the implementations of insertion sort and merge sort on the same machine. For input of size 'n', insertion sort runs in 8n2 steps, while merge sort runs in 64nlogn steps. For which values of 'n' does insertion sort beat merge sort? (Note: Finding the range of 'n' values) 02

Ans:

Calculate: It is obvious that insertion sort runs at quadratic time which is definitely worse than merge sort's linearithmic time for very large values of n. We know for $n = 1$, merge sort beats insertion sort. But for values greater than that, insertion sort beats merge sort. So, we will start checking from $n = 2$ and go up to see for what value of n merge sort again starts to beat insertion sort.

Notice that for $n < 8$, $2^{n/8}$ will be a fraction. So, let's start with $n = 8$ and check for values of n which are power of 2.

$n = 8 \implies 2^{8/8} = 2^1 = 2 < 8$ $\quad n = 16 \implies 2^{16/8} = 2^2 = 4 < 16$ $\quad n = 32 \implies 2^{32/8} = 2^4 = 16 < 32$
$n = 64 \implies 2^{64/8} = 2^8 = 256 > 64$

Somewhere between 32 and 64, merge sort starts to beat insertion sort. Let's do what we were doing but now we are going to try middle value of either range, repeatedly (binary search).

$n = \frac{32+64}{2} = 48 \implies 2^{48/8} = 64 > 48$ $\quad n = \frac{32+48}{2} = 40 \implies 2^{40/8} = 32 < 40$
$n = \frac{40+48}{2} = 44 \implies 2^{44/8} = 44.8 > 44$ $\quad n = \frac{40+44}{2} = 42 \implies 2^{42/8} = 38.4 < 42$
$n = \frac{42+44}{2} = 43 \implies 2^{43/8} = 42.4 < 43$

So, at $n = 44$, merge sort starts to beat insertion sort again. Therefore, for $1 < n < 44$, insertion sort beats merge sort.

A8 Identify the running time of the following functions. 02

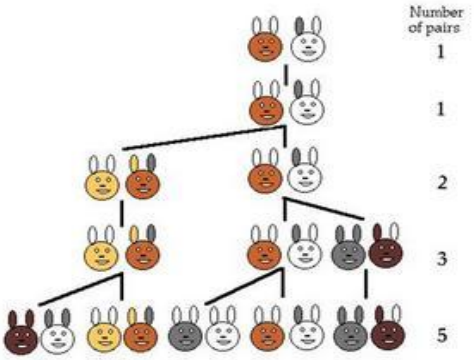
```
void function1(int n)
{
  int i = 0;
  if (n > 1)
       fun1(n-1);
  for (i = 0; i < n; i++)
      printf(" * ");
}
```

```
void function2(int n)
{
   int i = 1, s =1;
   while(s <= n)
   {
        i++;
        s + = i;
        printf("*");
   }    }
```

Ans: Function1: T(n)=T(n-1)+n, T(1)=1
$O(n^2)$
Function 2: O(logn)

A9 A man puts a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair which from the second month on becomes productive? Formulate a recurrence to find the number of pairs of rabbits produced. [Use diagrammatic representation to find the pairs of rabbits] 02

Ans: At the end of one year there will be 233 pairs.
T(n)=T(n-1)+T(n-2), T(0)=0,T(1)=0



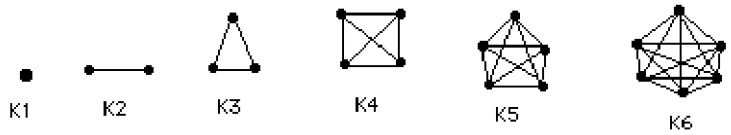
A10 Write the recurrence and analyze the number of times the basic operation is executed. 02

```
int DoSomething(int n)
{
    if (n<=2)
        return 1;
    else
        return DoSomething(sqrt(n)) + n
}
```

Ans: $T(n)=T(\sqrt{n})+1$, $T(2)=0$
Sub $n=2^m$
$T(2^m)=T(2^{m/2})+1$
In terms of S(m), $S(m)=S(m/2)+1$
Solving this we get, $S(m)=\log_2 m$
Again converting m to n, we get time complexity as log logn

Answer All Questions PART- B (5 x 16 = 80 Marks)

B1a)i) Let K_n be a complete graph on 'n vertices (i.e., between any two of the n vertices lies an edge). 04
Define a recurrence relation for the number of edges in the given complete graph K_n, and then solve this by using substitution method.



ANS: $T(1)=0$, $T(n)=T(n-1)+n-1$.....(1)
Sub n=n-1 in (1)
$T(n-1)=T(n-2)+n-2$.....(2)
Sub (2) in (1)
$T(n)=T(n-2)+n-2+n-1$
Generally, $T(n)=T(n-i)+n-i+n-(i-1)+.....+n-1$
Sub $n-i=1 \Rightarrow i=n-1$
$T(n) = T(1)+1+2+3+....+n-1$
$= 0+1+2+3+4.....+n-1$
$= n(n-1)/2$

B1a)ii) Solve the following recurrence using substitution method and find its asymptotic time complexity. 04
Recurrence relation: $T(0) = 5$
$T(N)= NT(N-1)$

ANS: $T(n)=n.T(n-1)$.....(1)
Sub n=n-1 in (1)
$T(n-1)=n-1T(n-2)$.....(2)
Sub (2) in (1)
$T(n)=n*n-1*T(n-2)$
$T(n)=T(n-2)*n-1*n$
:
:
$T(n)=T(n-i)*n-(i-1)*n-(i-2)*.....*n$
$n-i=0 \Rightarrow i=n$
$T(n)=T(0)*1*2*3*4*....*n$
$=5*n!$

B1a)iii) Given an array of size N consisting of only 0's and 1's, which is sorted in such a manner that all the 1's are placed first and then they are followed by all the 0's. You have to find 08

the count of all the 0's.

Input:

- The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. The first line of each test case contains an integer N, where N is the size of the array A[ ].
- The second line of each test case contains N space separated integers of all 1's followed by all the 0's, denoting elements of the array A[ ].

Output:

- Print the number of 0's in the array.

Expected Complexity: O(log N) [partial credit can be given for solution in higher complexities]

Input:

3
12
1 1 1 1 1 1 1 1 1 0 0 0
5
0 0 0 0 0
6
1 1 1 1 1 1

Output:

3
5
0

ANS:

```
int firstZero(int arr[], int low, int high) {
   if (high >= low)     {
      int mid = low + (high - low)/2;
      if (( mid == 0 || arr[mid-1] == 1) && arr[mid] == 0)
         return mid;
       if (arr[mid] == 1)
         return firstZero(arr, (mid + 1), high);
      else
         return firstZero(arr, low, (mid -1));
   }
   return -1;
}
int countZeroes(int arr[], int n) {
      int first = firstZero(arr, 0, n-1);
   if (first == -1)
      return 0;

   return (n - first);
}
```

(OR)

B1b)i) Write a recursive algorithm to Check whether a given String is Palindrome or not. 06

ANS:

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
bool isPalRec(char str[], int s, int e) {
   if (s == e)
      return true;
   if (str[s] != str[e])
      return false;
```

```
    if (s < e+1)
        return isPalRec(str, s+1, e-1);
return true;
}
bool isPalindrome(char str[]) {
    int n = strlen(str);
    if (n == 0)
    return true;
    return isPalRec(str, 0, n-1);
}
```

B1b)ii) Each of 'n' users spends some time on a social media site. For each i=1…..n, user i enters the site at time ai and leaves at time $b_i >= a_i$. You are interested in the question: how many distinct pairs of users are ever on the site at the same time? (Here, the pair (i; j) is the same as the pair (j; i)). Example: Suppose there are 5 users with the following entering and leaving times: 10

Then, the number of distinct pairs of users who are on the site at the same time is three: these pairs are (1; 2), (4; 5), (3; 5).

(a) Given input (a_1,b_1); (a_2,b_2)…….(a_n,b_n) as above, there is a straightforward algorithm that takes about $O(n^2)$ time to compute the number of pairs of users who are ever on the site at the same time. Give this algorithm and explain why it takes time about n^2.

(b) Propose an O(nlog(n))-time Greedy algorithm to do the same task.

User	Enter time	Leave time
1	1	4
2	2	5
3	7	8
4	9	10
5	6	10

ANS:

1. printf("The final activity schedule is ::\n");

```
for(i=0;i<n;i++)    {
for(j=i+1;j<n;j++)    {
  if(f[i]>s[j])
    printf("[%d],[%d]\t",a[i],a[j]);
} }
 printf("\n");
 return 0;
}
```

2. At first, sort the given values based on the starting time. This sorting takes n log n as time complexity.

Pseudo Code:

```
printf("The final activity schedule is ::\n");
for(i=0;i<n;i++)    {
for(j=i+1;j<n;j++)    {
  if(f[i]>s[j])
    printf("[%d],[%d]\t",a[i],a[j]);
   else
      break;
} }
 printf("\n");
```

```
    return 0;
}
```

B2a)i) Insert() and DeleteMin() are two fundamental operations of priority queues. What is the worst case time cost of these two operations in the case where the priority queues is implemented using the following data structures? 04

1. Linked list 2. Sorted linked list 3. Binary search tree 4. Binary heap

ANS: 1) Linked List: Insert() – O(n) ; DeleteMin() – O(n)
2) Sorted Linked List: Insert(n) – O() ; DeleteMin() – O(1)
3) Binary Search Tree: Insert(n) – O() ; DeleteMin() – O(n)
4) Binary Heap: Insert() – O(log n) ; DeleteMin() – O(1)

B2a)ii) Given two numbers N1 and N2 that are separated by a space, we need to swap first digit of first number with last digit of second number and last digit of first number with the first digit of second number. 12

Input:

The first line of the input contains a single integer T, denoting the number of test cases. Then T test cases follow. Each test case has a single line input where two integers N1 and N2, which are separated by a space, are inputted.

Output:

Print two numbers separated by a space such that first digit of first number has been swapped with last digit of second number and last digit of first number has been swapped with the first digit of second number.

Input:

5
1234 5678
8903 4863
7 8
12 21
666 999

Output:

8235 4671
3904 3868
8 7
12 21
969 696

Explanation:

Test case 1: 1234 5678
[1]23(4) (5)67[8]
| |_SWAP_| |
|_____SWAP____|
Hence we get 8235 4671 as the answer

ANS:

```
void swap(int a, int b) {
int x,y,len1=0,len2=0,n1,n2;
x=a;
y=b;
while(a!=0) {
a=a/10;
len1=len1+1;
}
while(b!=0) {
b=b/10;
len2=len2+1;
```

```
}
printf("%d%d",len1,len2);
n1=y%10*expo(10,len1-1)+x%expo(10,len1-1);
n2=x%10*expo(10,len2-1)+y%expo(10,len2-1);
n1=n1/10;
n2=n2/10;
n1=n1*10+y/expo(10,len2-1);
n2=n2*10+x/expo(10,len1-1);
printf("The nos are:%d\t%d",n1,n2);
}
```

(OR)

B2b)i) Given a String of length N, write an algorithm to reverse the words in it. Words are separated by dots. 08

Input:

The first line contains T denoting the number of test cases. Then follows description of test cases. Each case contains a string containing spaces and characters.

Output:

For each test case, output a single line containing the reversed String.

Input:

2
i.like.this.program.very.much
pqr.mno

Output:

much.very.program.this.like.i
mno.pqr

ANS:

```
#include<stdio.h>
void reverse(char *begin, char *end);
void reverseWords(char *s) {
  char *word_begin = s;
  char *temp = s;
  while( *temp )   {
    temp++;
    if (*temp == '\0')
        reverse(word_begin, temp-1);
    else if(*temp == ' ')      {
      reverse(word_begin, temp-1);
      word_begin = temp+1;
}}
  reverse(s, temp-1);
}
void reverse(char *begin, char *end) {
  char temp;
  while (begin < end)   {
    temp = *begin;
    *begin++ = *end;
    *end-- = temp;
}}
```

B2b)ii) An array element is peak if it is NOT smaller than its neighbors. For corner elements, we need to consider only one neighbor. For example, for input array {5, 10, 20, 15}, 20 is the only peak element. For input array {10, 20, 15, 2, 23, 90 and 67}, there are two peak elements: 20 and 90. Note that we need to return any one peak element. Use divide and conquer method. Complete the following algorithm to find a peak element in it. 08

```
/ A program to find a peak element using divide and conquer
#include <stdio.h>
// A binary search based function that returns index of a peak element
int findPeakUtil(int arr[], int low, int high, int n)
{
   // Find index of middle element
   int mid = _____.
   // Compare middle element with its neighbours (if neighbours exist)
   if ((mid = = 0 || _____) && (mid == n-1 || _____))
            return mid;
   // If middle element is not peak and its left neighbour is greater than it, then left half
must have a peak element
   else if (_____ && _____)
      return findPeakUtil(arr, _____,_____, n);
  // If middle element is not peak and its right neighbour is greater than it, then right half
must have a peak element
   else return findPeakUtil(arr,_____, _____, n);
}
int findPeak(int arr[], int n)
{
   return findPeakUtil(arr, 0, n-1, n);
}
int main()
{
   int arr[] = {1, 3, 20, 4, 1, 0};
   int n = sizeof(arr)/sizeof(arr[0]);
   printf("Index of a peak point is %d", findPeak(arr, n));
   return 0;
}
```

ANS:

```
#include <stdio.h>
 int findPeakUtil(int arr[], int low, int high, int n) {
   int mid = low + (high - low)/2;  /* (low + high)/2
   if ((mid == 0 || arr[mid-1] <= arr[mid]) &&
      (mid == n-1 || arr[mid+1] <= arr[mid]))
         return mid;
   else if (mid > 0 && arr[mid-1] > arr[mid])
      return findPeakUtil(arr, low, (mid -1), n);
   else return findPeakUtil(arr, (mid + 1), high, n);
}
int findPeak(int arr[], int n) {
   return findPeakUtil(arr, 0, n-1, n);
}
int main()  {
   int arr[] = {1, 3, 20, 4, 1, 0};
   int n = sizeof(arr)/sizeof(arr[0]);
   printf("Index of a peak point is %d", findPeak(arr, n));
   return 0;
}
```

B3a)i) Given an n x n matrix and a number x, find position of x in the matrix if it is present in it. Else print “Not Found”. In the given matrix, every row and column is sorted in increasing order. 12

Input : mat[4][4] = { {10, 20, 30, 40},
{15, 25, 35, 45},
{27, 29, 37, 48},
{32, 33, 39, 50} };
Key Element x = 29 Output : Found at (3, 2)
Input : mat[4][4] = { {10, 20, 30, 40}, {15, 25, 35, 45},
{27, 29, 37, 48}, {32, 33, 39, 50}};
Key Element x = 100 Output : Element not found

1) Write a Brute force algorithm for the given problem and state its time complexity. (Note: Pseudo Code should not exceed 15 lines)
2) Design an algorithm using divide and conquer strategy and it should have linear time complexity.

ANS: (Note: Pseudo Code should not exceed 10 lines)

1) Brute Force Algorithm:

```
void search(int mat1[][N], int x) {
    int i, j, flag=0;
    for (i = 0; i < N; i++)     {
        for (j = 0; j < N; j++)       {
            if ( mat[i][j] == x )       {
                    printf("n Found at %d, %d", i, j);
                    flag=1;
} }
     if(flag==0)
         printf("\n Element not found");
}
```

2) Divide and Conquer Technique:

```
#include<stdio.h>
int search(int mat[4][4], int n, int x) {
  int i = 0, j = n-1; //set indexes for top right element
  while ( i < n && j >= 0 )   {
    if ( mat[i][j] == x )      {
      printf("n Found at %d, %d", i, j);
      return 1;
     }
    if ( mat[i][j] > x )
      j--;
    else  //if mat[i][j] < x
      i++;
   }
printf("\n Element not found");
return 0; // if ( i==n || j== -1 )
}
```

B3a)ii) You wish to store a set of ‘n’ numbers in memory. For each application below, state the appropriate data structure that suits better. Explain your answer briefly. 04

1) Want to find the maximum element quickly.
2) Want to be able to delete an element quickly.
3) Want to be able to form the structure quickly.
4) Want to find the minimum element quickly.

ANS:

1) Priority Queue or Max-Heap
2) Linked List
3) Graph
4) Priority Queue or Min-Heap

(OR)

B3b)i) For each of the following problems, give an algorithm that finds the desired numbers within the given amount of time. 12

For example, given an array S[]={6,13,19,3,8},
Here pair (19,3) maximizes the difference, while (8,6) minimizes the difference.

(a) Let S be an unsorted array of n integers. Give an algorithm that finds the pair x,y∈ S that maximizes |x−y|. Your algorithm must run in O(n) worst-case time.

(b) Let S be a sorted array of n integers. Give an algorithm that finds the pair x,y∈ S that maximizes |x−y|. Your algorithm must run in O(1) worst-case time.
(c) Let S be an unsorted array of n integers. Give an algorithm that finds the pair x,y∈ S that minimizes |x−y|, for x≠y. Your algorithm must run in O(nlogn) worst-case time.

(d) Let S be a sorted array of n integers. Give an algorithm that finds the pair x,y∈ S that minimizes |x−y|, for x≠y. Your algorithm must run in O(n) worst-case time.

ANS: (a) Iterate over the array once, keeping track of the max and min values respectively for x and y. This will have $O(n)$ worst-case time.

(b) The answer is straight forward: $S[1]$ and $S[n]$, since they are the two extreme values.

(c) Sort the array with any $n\log(n)$ method. Then scan through the sorted array to find the smallest gap, thus the desired pair.

(d) Same as above.

B3b)ii) For each of the following pairs of functions, determine whether f(n) is in O(g(n)), f(n) is in Ω(g(n)), or f(n)=Θ(g(n)). Briefly state the reason behind it. 04

1. $f(n) = \log n^2 \;\; ; g(n) = \log n \;\; + 5$
2. $f(n) = \sqrt{n} \;\; ; g(n) = \log n^2$
3. $f(n) = \log^2 n \;\; ; g(n) = \log n$
4. $f(n) = n \;\; ; g(n) = \log^2 n$
5. $f(n) = n \log n + n \;\; ; g(n) = \log n$
6. $f(n) = 10 \;\; ; g(n) = \log 10$
7. $f(n) = 2^n \;\; ; g(n) = 10n^2$
8. $f(n) = 2^n \;\; ; g(n) = 3^n$

ANS:
1. $\log n^2 = \Theta(\log n + 5)$
2. $f(n) = \Omega(g(n))$
3. $f(n) = \Omega(g(n))$
4. $f(n) = \Omega(g(n))$
5. $f(n) = \Omega(g(n))$
6. $f(n) = \Theta(g(n))$
7. $f(n) = \Omega(g(n))$
8. $f(n) = O(g(n))$

B4a)i) The following are the starting and ending times of activities A, B, C, D, E, F, G and H respectively in chronological order: “as, bs, cs, ae, ds, ce, es, fs, be, de, gs, ee, fe, hs, ge, he”. Here, *xs* denotes the starting time and *xe* denotes the ending time of activity ‘*x*’. The problem is to schedule the activities in a set of seminar hall rooms available to us. An activity can be scheduled in a seminar room only if the room is reserved for the activity 04

for its entire duration. What is the minimum number of rooms required? Use appropriate design strategy to solve the problem.

ANS:

Sequence	as	bs	cs	ae	ds	ce	es	fs	be	de	gs	ee	fe	hs	ge	he	
Rooms	0	1	2	3	2	3	2	3	4	3	2	3	2	1	2	1	0

Logic: Increase the number of rooms if some activity starts and decrease the number of rooms by 1 if one activity ends.

Hence the minimum number of rooms required at a time = 4

B4a)ii) A word 'W' is said to be beautiful if the following two conditions are satisfied: 12

Conditions

- No two consecutive characters are the same.
- No two consecutive characters are in the following vowel set: {a, e, i, o, u, y}. Note that we consider 'y' to be a vowel in this challenge.

For example:

Input 1: W=Batman Output 1 : Returns Yes
Input 2: W=Apple Output 2: Returns No
Input 3: W= Beauty Output 3: Returns No

The string batman is beautiful because it satisfies the given criteria; however, apple has two consecutive occurrences of the same letter (pp) and beauty has three consecutive vowels (eau), so those words are not beautiful. Given 'W' print Yes if it is beautiful or No if it is not.

a) Discuss a brute force algorithm for the given problem.
b) Analyze the running time of the solution.

ANS: a) Algorithm:

```
#include<stdio.h>
#include<string.h>
void main() {
        int i,j,same=0;
        char arr[100];
        printf("String: ");
        scanf("%s",arr);
        for (i=0;i<strlen(arr);i++)  {
        if (arr[i]=='a'||arr[i]=='e'||arr[i]=='i'||arr[i]=='o'||arr[i]=='u'||arr[i]=='y')
                arr[i] = '#';
        }
        for (j=0;j<strlen(arr)-1;j++) {
                if (arr[j]==arr[j+1]) {
                        same=1;
                        break;
                }
        }
        if (same==1)
                printf("No\n");
        else
                printf("Yes\n");
}
```

b) Running Time: O(n)

(OR)

B4b)i) Suppose you are given an array A of n sorted numbers that has been circularly shifted k positions to the right. For example, {35,42,5,15,27,29} is a sorted array that has been circularly shifted k=2 positions, while {27,29,35,42,5,15} has been shifted k=4 positions. 10

1) Suppose you know what k is. Give an O(1) algorithm to find the largest number in A.
2) Suppose you do not know what k is. Give an O(log n) divide and conquer based algorithm to find the transition point and then use O(1) algorithm to find the largest number in A. For partial credit, you may give an O(n) algorithm.

Note that the value of the shift is the index of the largest element. Hence, it suffices to find the index of the largest element.

ANS: 1. Since set is sorted the max element will lie at position
Since set is sorted the max element will lie at position
Max = Set[k] where k!= 0
Set[n] where K == n

2. Apply binary search to find out transition point
Assume set indexes are zero based

```
FindNumberOfRotations(A):
1. if  (A[0] < A[n-1]) then
   return 0//There is no rotation made
2. low =  0, high =1
3. mid = (low + high)/2
4. if(A[mid] < A[high]) then
      //Transition lies in left half of the array
      if A[mid-1] > A[mid] then return mid
      else
         high = mid-1
         Go to step 3.
   else
      //Transition lies in right half of array
      if(A[mid] > A[mid+1]) then return mid+1
      else
         low = mid+1
         Go to step 3
```

B4b)ii) Consider the modified binary search algorithm so that it splits the input not into two sets of almost-equal sizes, but into three sets of sizes approximately one-third. 06

i) Design a recursive algorithm for this ternary search.
ii) Write down the recurrence for this ternary search algorithm to find the asymptotic complexity of the designed algorithm.

ANS: The recurrence for normal binary search is $\mathbf{T_2(n) = T_2(n/2)+1}$.
This accounts for one comparison (on an element which partitions the n-element list of sorted keys into two n 2 -element sets) and then a recursive call on the appropriate partition. For ternary search, two comparisons are made on elements which partition the list into three sections with roughly n/3 elements and recurse on the appropriate partition. Thus analogously, the recurrence for the number of comparisons made by this ternary search is:

$$\mathbf{T_3(n) = T_3(n/3) + 2.}$$

However, just as for binary search the second case of the Master Theorem applies.
Therefore, $\mathbf{T_3(n) \in \Theta(\log(n)).}$

B5a)i) Steve has a string, S, consisting of 'n' lowercase English alphabetic letters. In one operation, he can delete any pair of adjacent letters with same value. For example, string 08

"aabcc" would become either "aab" or "bcc" after operation.
Steve wants to reduce as much as possible. To do this, he will repeat the above operation as many times as it can be performed. Help Steve out by finding and printing S's non-reducible form!
Note: If the final string is empty, print “Empty String”.
Input Format
A single string S.
Constraints: 1<=n<=100
Output Format: If the final string is empty, print “Empty String” otherwise, print the final non-reducible string.
Sample Input 1: baab Sample Output 1: Empty String
Explanation 1: Steve can perform the following sequence of operations to get the final string:

1. baab → bb
2. bb → Empty String

Thus, we print Empty String.
Sample Input 2: aa Sample Output 2: Empty String
Explanation 2: Steve can perform the following sequence of operations to get the final string:

1. aa → Empty String

Thus, we print Empty String.
Sample Input 2: aaabccddd Sample Output 2: abd
Sample Case 2: Steve can perform the following sequence of operations to get the final string:

1. aaabccddd → abccddd
2. abccddd → abddd
3. abddd → abd

Thus, we print abd.

i) Design an algorithm for the above problem.
ii) Implement the above and state its time complexity.[Use any programming language]

ANS:

```
#include<stdio.h>
char s[1000];
int main()
{
	printf("Enter the string: ");
	scanf("%s", s);
	int k = 1,i;
	for (i = 1; s[i]; i++) {
		if (s[i] == s[k - 1]) k--;
		else s[k++] = s[i];
	}
	s[k] = 0;
	if (k == 0) printf("Empty String\n");
	else printf("%s\n", s);
	return 0;
}
```

B5a)ii) Answer the following questions based on various Sorting and Searching techniques. 08

i) You have a computer with only 2Mb of main memory. How do you use it to sort a large file of 500 Mb that is on disk?

ii) _____ sorting techniques are applied when the entire collection of data to be sorted is small enough that the sorting can take place within main memory.

iii) ______sorting algorithm is frequently used when ‘n is small where n is total number of elements and is almost sorted.

iv) _____ sorting algorithm is efficient and frequently used when ‘n is small where n is total number of elements and the list is random.

v) The k^{th} smallest element can be efficiently found using _____ sorting technique.

vi) State atleast 4 sorting techniques that runs in O(nlogn)

vii) Consider a situation where swap operation is very costly. Which sorting algorithm is most efficient and is preferred so that the numbers of swap operations are minimized in general?

viii) Which sorting algorithm is the most efficient to sort string consisting of ASCII characters?

ANS: i) Merge Sort
ii) Internal Sorting
iii) Insertion Sort
iv) Counting Sort
v) Min Heap
vi) Merge Sort, Heap Sort, Tournament Sort and Quick Sort.
Vii) Selection Sort
viii) Counting Sort

(OR)

B5b)i) There are N trees in a circle. Each tree has a fruit value associated with it. 16

A bird has to sit on a tree for 0.5 sec to gather all the fruits present on the tree and then the bird can move to a neighboring tree. It takes the bird 0.5 seconds to move from one tree to another. Once all the fruits are picked from a particular tree, she can’t pick any more fruits from that tree. The maximum number of fruits she can gather is infinite.

We are given N and M (the total number of seconds the bird has), and the fruit values of the trees. We have to maximize the total fruit value that the bird can gather. The bird can start from any tree.

Input:

The first line contains a positive integer T denoting the number of test cases. Each of the test case consists of 2 lines. The first line of each test case contains two integers N and M. Next line contains space separated sequence of N integers denoting the fruit value of each tree.

Output:

For each test case, print in a new line, the maximum number of fruits the bird can gather. As the results can be large, print your answer modulo 10^9+7

2
7 3
2 1 3 5 0 1 4
6 2
1 6 2 5 3 4

Output:

9
8

Explanation:

Assume 0-based indexing.
For the 1st case:
She can start from tree 1 and move to tree2 and then to tree 3. Hence, total number of gathered fruits = 1 + 3 + 5 = 9.
For the 2nd case:
She can start from tree 1 and move to tree 2. In this case she will gather 6 + 2 = 8 fruits.

She can also start from tree 3 and move to tree 4. In this case, too, she will gather 5 + 3 = 8 fruits.

ANS:

```
#include <stdio.h>
#define MAX 100000
int main() {
   int t;
   printf("Cases: ");
   scanf("%d",&t);
   while(t--)    {
      int m,n,i,j,asum=0,sum=0;
      int a[MAX];
      scanf("%d",&n);
      scanf("%d",&m);
      printf("Array Elements: ");
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);
      if(m>n)
      m=n;
      for(i=0;i<n;i++)        {
         asum=0;
         for(j=0;j<m;j++)        {
          asum+=a[(i+j)%n];
       }
       if(asum>sum)
       sum=asum;
   }
   printf("Result = %d\n",sum);
   }
         return 0;
}
```

Name & Signature of the Faculty In-Charge

Signature of HoD/Director