COL100: II semester, 2014-15.
Minor II
8AM to 9AM, 20th March 2015

Question	1 (Max 6)	2 (Max 2)	3 (Max 5)	4 (Max 7)	Total (Max 20)
Marks	3	0	5	7	15

Note: All programs are to be written in C++. However, minor mistakes in C++ syntax will be ignored and **no marks will be deducted**.

Scanned by CamScanner

Name:_____ Entry No:_____ Gp No:____

Q1. Consider the following functions. You are required to give the results corresponding to the function calls as asked below for the respective functions. These function calls are made from the `main` function.

Q1.1 (3 marks) What are the results of the calls `fun1(2,64)` and `fun1(7,84)`? What is the relation between the parameters m and n when the function returns `true`?

Answer:

```
bool fun1(int m, int n)
{
   if (m>n) return false;
   else {
     while (n >= m) {
       if (n%m == 0) n=n/m;
       else return false;
     }
     if (n == 1) return true;
     else return false;
   }
}
```

① When fun1(2, 64) true

② When fun1(7, 84) false.

(3) When it returns true, it means n is in terms of m^a i.e. $n = m^a$ where a is any positive integer

Q1.2 (3 marks) What is the result of the call `fun2(B,0,5,6)`, where B is an array of size 6 and initialised as `B[6] = {11,5,3,15,10,18}`? State in one sentence in English about what the function finds.

Answer:

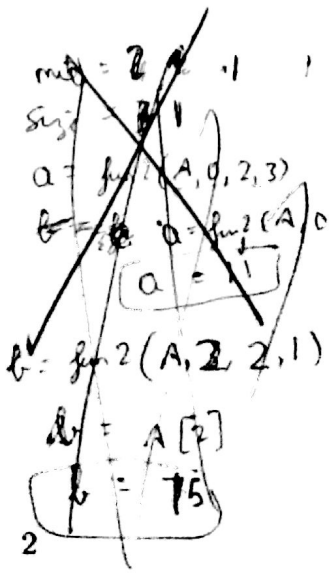
B[] 0 ... 6

```
int fun2(int A[], int left, int right, int size)
{
   int a, b, mid;

   if (size == 1) return A[left];
   else
   {
     mid = left + (right - left)/2;
     size = size/2;
     a = fun2(A,left,mid,size);
     b = fun2(A,mid+1,right,size);
     if (a>b) return a;
     else return b;
   }
}
```

Result : 15 × 18

It divides array B in 2 parts & give the highest B first & finds largest element in array



Scanned by CamScanner

Q2 (2 marks) Rank from 1 to 4 indicating better to worse the order of time complexities given below using big O notation. Rank 1 will be of the one which has the least time complexity.

Answer

- $O(n^{10})$ — 4
- $O(2^n)$ — 3
- $O(\sqrt{n})$ — 1
- $O(\log_2 n)$ — 2

Q3. (5 marks) A palindrome number is a number such that if we reverse it, it does not change. For example, 121, 212, 12321 are palindrome numbers. A function below is to be completed that returns `true` if the number n is palindrome otherwise it returns `false`. Find the time complexity of the function using big O notation in terms of n.

Answer:

```
bool palindrome(int n)
{
   int a,i, b=0 ;
   a=n;
   for (i=0; a!=0 ; i++)
   {
      b= b*10+ a%10;
      a= a/10 ;
   }
   if ( b==n)
      return true ;
   else
      return false;
}
```

C = No. of digits in n

O(2c)

5

Scanned by CamScanner

Name:_____ Entry No:_____ Gp No:____

$A A^T = I$

Q4. (7 marks) A square matrix NxN is orthogonal if the product of the matrix A and its transpose A^T is an Identity matrix I. Complete the program below for checking whether a matrix is an orthogonal matrix or not. Please note that an identity matrix of size 2x2 is $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ You can declare additional scalar variables if need be. As indicated in the comments the running time complexities of the parts for finding transpose, multiplication of matrices and checking whether the matrix is identity or not must be $O(N^2)$, $O(N^3)$, and $O(N^2)$ respectively.

Answer:

```
#include <iostream>
using namespace std;

#define N 10          /* This provides the value of 10 to N which is used for
                         the size of the arrays */

int main ( )
{
      int A[N][N], AT[N][N], AAT[N][N];
      int i, j;
      bool identity = false;

      /* The matrix A is read into by an appropriate read function. You are not
      required to write the code for reading the matrix */

      /* Below is the part of the code which finds the transpose matrix AT of
      the matrix A. Complete the code. The running time complexity of the code
      below should be O(N^2) i.e., quadratic in N*/

      for (i=0; i<N; i++) {
       for (j=0; j<N; j++) {
```

AT[i][j] = A[j][i] (2) //AT is transpose matrix

Scanned by CamScanner

```
}

/* Below is the part of the code which finds the multiplication of the
matrix AT and the matrix A and puts the result in the matrix AAT.
Complete the code. The running time complexity of the code below
should be O(N^3) i.e., cubic in N*/

for (i=0; i<N; i++) {
 for (j=0; j<N; j++) {
    AAT[i][j] = 0;
    for (int k=0; k<N; k++)
    {
      AAT[i][j] += A[i][k] * AT[k][j];
    }

  }
}

/* Below is the part of the code which finds if the matrix AAT is an
identity matrix or not. Complete the code. The running time complexity
of the code below should be O(N^2) i.e., quadratic in N*/
  int identity = 1;
for (i=0; i<N; i++) {
 for (j=0; j<N; j++) {

      if (i != j)
      { if (AAT[i][j] != 0)
          identity = 0;
      }
      if (i == j)
      { if (AAT[i][j] != 1)
        identity = 0;
      }

if (identity) cout << "The given matrix is orthogonal" << endl;
else cout << "The given matrix is not orthogonal" << endl;
}
```

/* Even if a single element is non zero identity becomes 0 */

/* Even if a single diagonal element is not 1, identity becomes 0 */

// If identity ≠ 0 this runs.
// If identity = 0, this condition runs.

Scanned by CamScanner