

DS.

unsigned int array **primefactors[]**. **num** contains the number to be processed by any of the described functions and **ans** contains the result generated by any of the functions except for **primeFactors()**, for which the result is put into **primefactors[]** array.

The class should be named "NumberTheoretic" and all principles of abstraction and encapsulation must be incorporated in writing it. You are at freedom to define variables and functions other than those described here.

Question No. 4

The following is C++ implementation for the naive prime factorization algorithm:

```
int input, factor, factor_remainder, check=0;
cout<<"\n\nEnter a number to get its prime factors: ";
cin>>input;
cout<<"\n\nThe prime factors of "<<input<<" are:";

factor=2;                // initialising factor with 2
                        // which could be the first factor

while(factor<=input)      // Termination Condition: factor is
{                          // increasing by 1 and input is reducing
                        // due to division

    if(isprime(factor)==1) // which means factor is prime
    {
        factor_remainder=input%factor;

        // If factor is prime we have two cases:

        //1) That factor divides the input completely
        if(factor_remainder==0)
        {
            input=input/factor;
            cout<<factor<<" ";
        }

        //2) The factor does not divide the input
        if(factor_remainder!=0)
        {
            factor=factor+1;
        }
    }
    else                  // factor is composite
    {
        factor=factor+1;
    }
}
```

This program has an underlying problem: it is inefficient for some of the factors, as it tests "factor" for primality without consideration. Make this program a bit more considerate by modifying it in such a way that "factor" variable is only tested for primality when a new "factor" is selected. Also discuss how much performance gain would this modification bring to the program.

Question No. 5

Describe the Quick-Sort algorithm using an example and write a C++ function definition to implement it on a character array which starts at index *i* and ends at index *j*:

```
void quickSort(char array[], int i, int j);
```

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY
SECOND YEAR SPRING SEMESTER (ELECTRICAL ENGINEERING)
EXAMINATION 2016

BATCH 2014-15 & PREVIOUS BATCHES

Time: 3 Hours

Dated: 28-04-2016

Max. Marks: 60

DATA STRUCTURE & ALGORITHMS- EE-264

Instructions:

- 1) Attempt four (4) questions in all – all questions carry equal marks.
- 2) Answer the questions objectively in legible writing. Any missing information may be assumed and clearly mentioned.
- 3) Use pen to answer the questions and be precise and objective while writing codes. Careless syntax errors will be dealt with serious deduction in marks.
- 4) Your algorithms/programs should be valid for all possible sets of inputs.

Question No. 1

a) What is "Algorithm Analysis" and why is it an important part of computer programming? Compare "Algorithm Analysis" with other desired characteristics of computer programming (compare with at-least five (05) other characteristics). (07)

b) Describe the three cases used to analyze algorithm performance, discussing significance of each method in detail. Demonstrate the application of ~~these~~ ^{any 1 of these} techniques on any algorithm of your choice and compute its asymptotic complexity with reference to $\theta(n)$ operator. (08)

Question No. 2

Using the C++ vector class and algorithm library, write a program that allows user to enter 100 names in a string vector (the user should be allowed to stop entering names intermediately), sort them in ascending order (alphabetical order), and finally display them on screen. (15)

After writing the program, discuss the method/algorithm used for sorting the sequence of strings. How different is it from sorting a number sequence?

Question No. 3

Write a C++ class that contains the following number theoretic algorithms as functions: (15)

- a) `bool isPrime(unsigned int N);` //checks whether N is prime or not
- b) `unsigned int gcd(unsigned int A, unsigned int B);` //computes greatest
//common divisor of A, and B integers
- c) `void primeFactors(unsigned int N, unsigned int factors[]);`
//computes prime factors of N and stores
//them in factors[] array
- d) `bool isEven(unsigned int N);` //checks if N is even or odd
- e) `bool isOdd(unsigned int N);` //checks if N is odd

Apart from these functions, there may be some general input/output functions in the class. The class needs to have public and private sections, moreover it must contain private data comprising of two unsigned integers **num** and **ans** and an

$$\theta(n) = n$$