

|  |  |
| --- | --- |
| **ASSIGNMENT** | |
| **Course Code** | 19CSC304A |
| **Course Name** | Operating Systems |
| **Programme** | B. Tech. |
| **Department** | Computer Science and Engineering |
| **Faculty** | Faculty of Engineering & Technology |

#### 

|  |  |
| --- | --- |
| **Name of the Student** | Tanishq R Porwar |
| **Reg. No** | 18ETCS002131 |
| **Semester/Year** | 5TH semester / 2018 batch |
| **Course Leader/s** | Ms. Naveeta |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Declaration Sheet** | | | | | | | | |
| Student Name | Tanishq R Porwar | | | | | | | |
| Reg. No | 18ETCS002131 | | | | | | | |
| Programme | B. Tech. | | | | | Semester/Year | 5th sem / 2018 batch | |
| Course Code | 19CSC304A | | | | | | | |
| Course Title | Operating Systems | | | | | | | |
| Course Date |  | | to | |  | | | |
| Course Leader | Ms. Naveeta | | | | | | | |
| **Declaration**  The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly. | | | | | | | | |
| Signature of the Student | |  | | | | | Date |  |
| Submission date stamp  (by Examination & Assessment Section) | |  | | | | | | |
| Signature of the Course Leader and date | | | | Signature of the Reviewer and date | | | | |
|  | | | |  | | | | |

# **Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[**Declaration Sheet** ii](#_Toc57678676)

[**Contents** iii](#_Toc57678677)

[Marking Scheme 4](#_Toc57678678)

[**Question No. 1** 6](#_Toc57678679)

[1.1 Introduction to multi-programming 6](#_Toc57678680)

[1.2 Effect of multi-programming on CPU utilisation 6](#_Toc57678681)

[**Question No. 2** 7](#_Toc57678682)

[2.1 Design and implementation of the application using sequential approach with functions 7](#_Toc57678683)

[2.2 Design and implementation of the application using multithreaded approach 7](#_Toc57678684)

[2.3 Comparison of the execution time of the above two versions of the program and its analysis 7](#_Toc57678685)

[**Question No. 3** 8](#_Toc57678686)

[3.1 Schedule of the processes using a Gantt chart 8](#_Toc57678687)

[3.2 Average waiting time and average turnaround time experienced 8](#_Toc57678688)

[3.3 Scheduling algorithm with better performance and its justification 8](#_Toc57678689)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assignment** | | | | |  | |  |  |
| Register No. | | | **18ETCS002131** | Name of Student |  | **Tanishq R Porwar** | | |
| **Sections** |  | Marking Scheme | | | **Max Marks** | | **First Examiner**    **Marks** | **Moderator**    **M**  **arks** |
| **Question 1** | Q1.1 | Introduction to multi-programming | | | 01 | |  |  |
| Q1.2 | Effect of multi-programming on CPU utilisation | | | 04 | |  |  |
|  | **Question 1 Max Marks** | | | **05** | |  |  |
| **Question 2** | Q2.1 | Design and implementation of the application using sequential approach with functions | | | 04 | |  |  |
| Q2.2 | Design and implementation of the application using multithreaded approach | | | 04 | |  |  |
| Q2.3 | Comparison of the execution time of the above two versions of the program and its analysis | | | 02 | |  |  |
|  | **Question 2 Max Marks** | | | **10** | |  |  |
| **Question 3** | Q3.1 | Schedule of the processes using a Gantt chart | | | 04 | |  |  |
| Q3.2 | Average waiting time and average turnaround time experienced | | | 04 | |  |  |
| Q3.3 | Scheduling algorithm with better performance and its justification | | | 02 | |  |  |
|  | **Question 3 Max Marks** | | | **10** | |  |  |
|  |  | **Total Assignment Marks** | | | **25** | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course Marks Tabulation** | | | | |
| **Component- 1(B) Assignment** | **First**  **Examiner** | **Remarks** | **Moderator** | **Remarks** |
| Q1 |  |  |  |  |
| Q2 |  |  |  |  |
| Q3 |  |  |  |  |
| **Marks (out of 25 )** |  |  |  |  |
| Signature of First Examiner Signature of Second Examiner | | | | |

# **Question No. 1**

**Solution to Question No. 1:**

## Introduction to multi-programming

In the early days, the role of the operating system was simple and straight-forward - load a program into memory and execute it via the CPU. In a modern computing system, there are usually several concurrent application processes which want to execute. Now it is the responsibility of the Operating System to manage all the processes effectively and efficiently.

Multiprogramming became possible when disks were introduced to the computing world. The concept of multiprogramming relies on the capability of a computer to store instructions/programs for long-term use. The goal is to reduce CPU idle time by allowing new jobs to take over the CPU whenever the currently running job needed to wait (e.g. for user I/O).

With the advent of multiprogramming, operating systems now faced different mechanics for program execution as multiple jobs now needed to be loaded into memory at the same time and several options existed for allocating CPU time. Two types of scheduling were introduced to handle this decision-making - jobscheduling and **CPU** scheduling. *Job scheduling* refers to the selection of jobs to load into memory. *CPU scheduling* refers to the selection of a job existing in memory to execute via the CPU. In a computer system, both these decisions are made by the operating system.

Scheduling Basics

Process scheduling is one of the most important functions of an operating system that supports multiprogramming. This function is heavily dependent on queues. There are three types of queues that are used in process scheduling:

* Job Queue - Contains all processes that have been introduced into the system
* Ready Queue - Contains processes that are waiting for CPU time, and can be selected to run at any time
* Device Queue - Contains processes waiting on a certain device. Each device has its own queue for processes that need input/output from it.

## 1.2 Effect of multi-programming on CPU utilisation

One of the most important aspects of an Operating System is to multi program.

In a computer system, there are multiple processes waiting to be executed, i.e. they are waiting when the CPU will be allocated to them and they begin their execution. These processes are also known as jobs. Now the main memory is too small to accommodate all of these processes or jobs into it. Thus, these processes are initially kept in an area called job pool. This job pool consists of all those processes awaiting allocation of main memory and CPU.

CPU selects one job out of all these waiting jobs, brings it from the job pool to main memory and starts executing it. The processor executes one job until it is interrupted by some external factor or it goes for an I/O task.

A multiprogramming is a parallel processing in which the multiple programs can run simultaneously. Most of the devices we use are uniprocessor like PC/Mobile/Tablet. Actually, Processor is programmed to use the scheduling. Scheduling is the process of arranging, controlling and optimising work, and workloads in a production process or manufacturing process. It creates and alias that all programs are working simultaneously, but in real all program/process works one at a time on the processor.

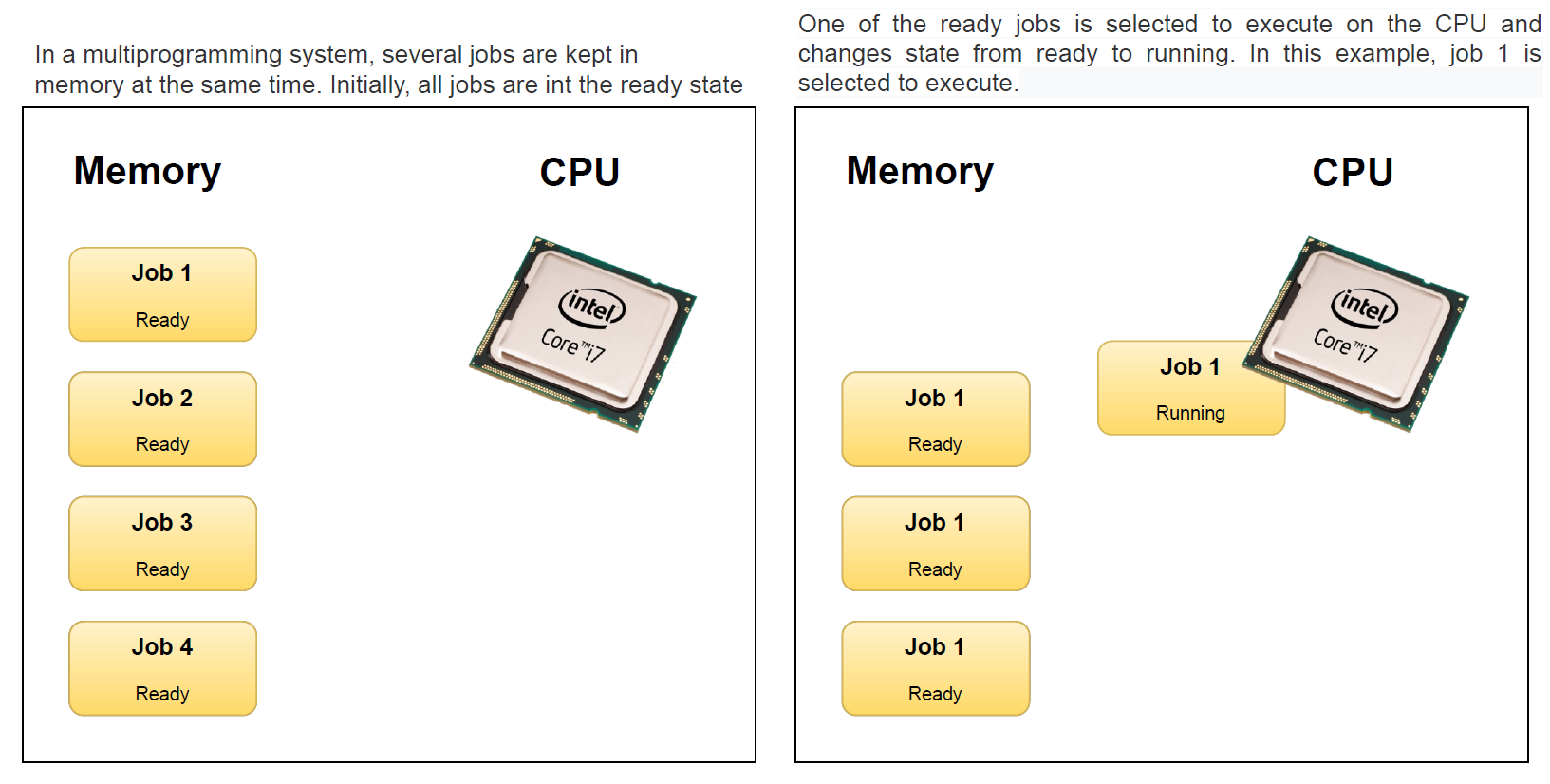
**Example:**Use a browser, play video, download apps and transfer data at the same time. In actual all processes are working one at a time on the processor. Switching between process/program is so fast that we never notice a difference.

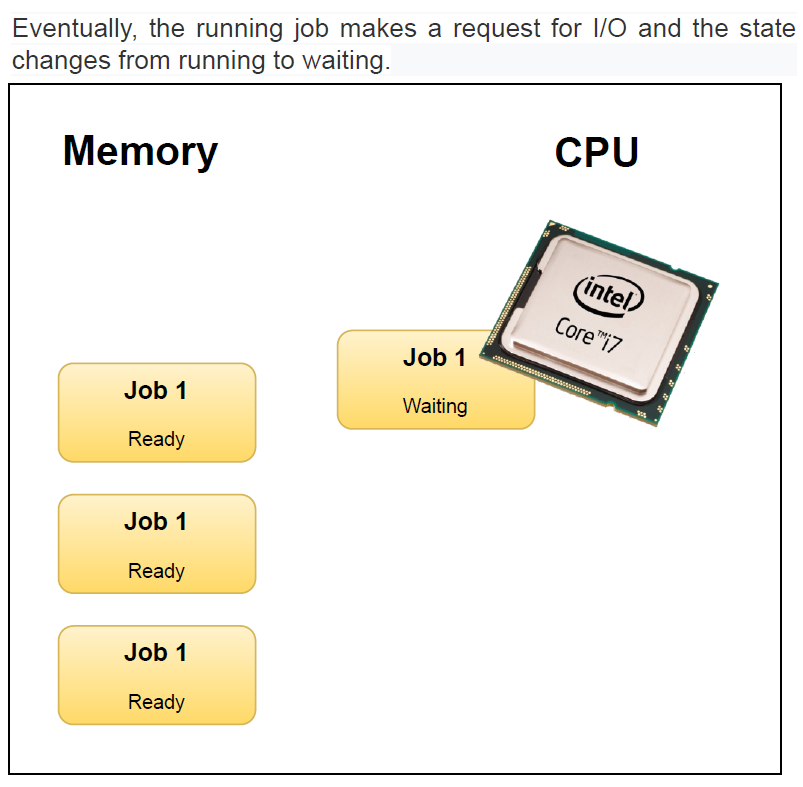
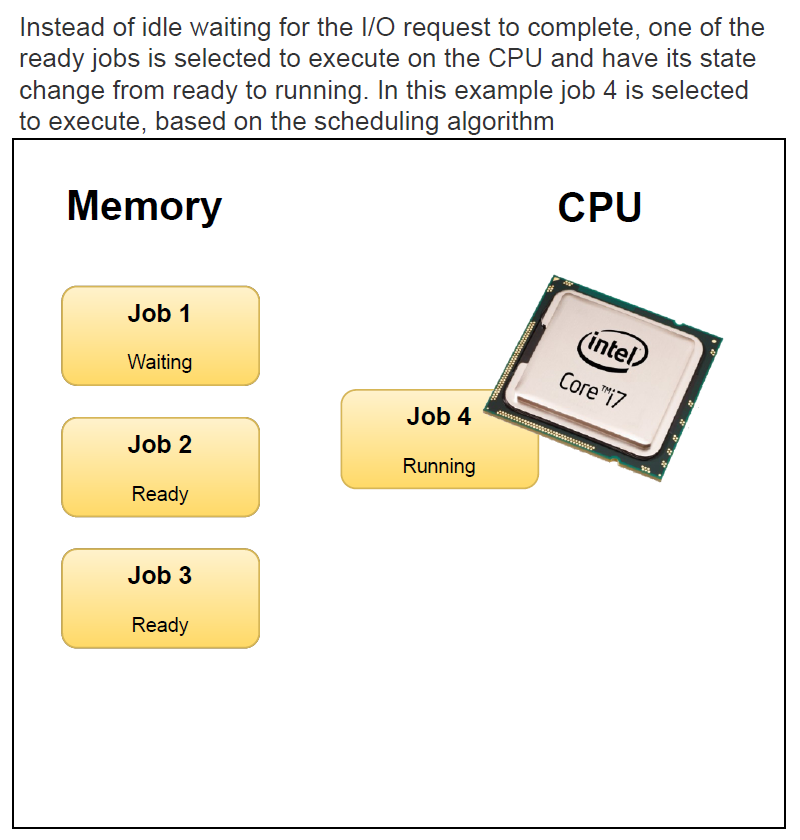
**Non-multi programmed system’s working –**

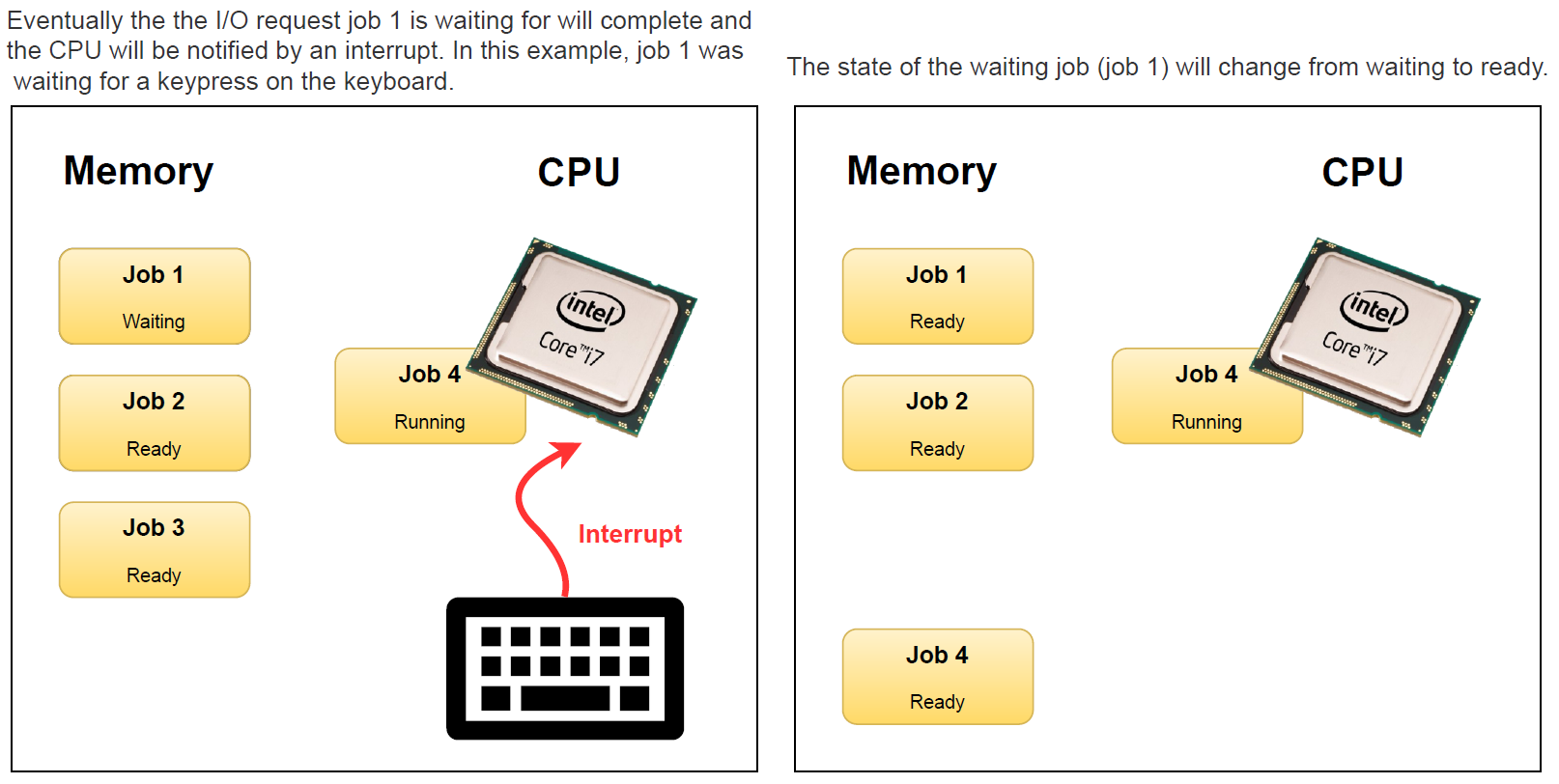
* In a non multi programmed system, as soon as one job leaves the CPU and goes for some other task (say I/O), the CPU becomes idle. The CPU keeps waiting and waiting until this job (which was executing earlier) comes back and resumes its execution with the CPU. So, CPU remains free for all this while.
* Now it has a drawback that the CPU remains idle for a very long period of time. Also, other jobs which are waiting to be executed might not get a chance to execute because the CPU is still allocated to the earlier job.  
  This poses a very serious problem that even though other jobs are ready to execute, CPU is not allocated to them as the CPU is allocated to a job which is not even utilizing it (as it is busy in I/O tasks).
* It cannot happen that one job is using the CPU for say 1 hour while the others have been waiting in the queue for 5 hours. To avoid situations like this and come up with efficient utilization of CPU, the concept of multi programming came up.

The main idea of multi programming is to maximize the CPU time.  
**Multi programmed system’s working –**

* In a multi-programmed system, as soon as one job goes for an I/O task, the Operating System interrupts that job, chooses another job from the job pool (waiting queue), gives CPU to this new job and starts its execution. The previous job keeps doing its I/O operation while this new job does CPU bound tasks. Now say the second job also goes for an I/O task, the CPU chooses a third job and starts executing it. As soon as a job completes its I/O operation and comes back for CPU tasks, the CPU is allocated to it.





* In this way, no CPU time is wasted by the system waiting for the I/O task to be completed.  
  Therefore, the ultimate goal of multi programming is to keep the CPU busy as long as there are processes ready to execute. This way, multiple programs can be executed on a single processor by executing a part of a program at one time, a part of another program after this, then a part of another program and so on, hence executing multiple programs. Hence, the CPU never remains idle.

# **Question No. 2**

**Solution to Question No. 2:**

## 2.1 Design and implementation of the application using sequential approach with functions

## 2.2 Design and implementation of the application using multithreaded approach

## 2.3 Comparison of the execution time of the above two versions of the program and its analysis

# **Question No. 3**

**Solution to Question No. 3:**

## 3.1 Schedule of the processes using a Gantt chart

**Non-Pre-emptive priority scheduling algorithm:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| processes | Burst time  (ns) | Arrival time  (ns) | Priority | Start  time | Completion time | Turnaround time | Waiting time | delay |
| 1 | 10 | 15 | 6 | 37 | 47 | 32 | 22 | 3.2 |
| 2 | 15 | 20 | 8 | 22 | 37 | 17 | 2 | 1.134 |
| 3 | 5 | 25 | 2 | 47 | 52 | 27 | 22 | 5.4 |
| 4 | 12 | 10 | 4 | 10 | 22 | 12 | 0 | 1 |

**Gantt chart for non-pre-emptive:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 4 | 2 | 1 | 3 |

0 10 22 37 47 52

Ready Queue: 4 2 1 3

**Pre-emptive priority scheduling algorithm:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| processes | Burst time  (ns) | Arrival time  (ns) | Priority | Start  time | Completion time | Turnaround time | Waiting time | delay |
| 1 | 10->5->0 | 15 | 6 | 15 | 40 | 25 | 15 | 2.5 |
| 2 | 15->10->0 | 20 | 8 | 20 | 35 | 15 | 0 | 1 |
| 3 | 5->0 | 25 | 2 | 47 | 52 | 27 | 22 | 5.4 |
| 4 | 12->7->0 | 10 | 4 | 10 | 47 | 37 | 25 | 3.08 |

**Gantt chart for pre-emptive:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 4 | 1 | 2 | 2 | 1 | 4 | 3 |

0 10 15 20 25 35 40 47 52

Ready Queue: 4 1 3 2

## 3.2 Average waiting time and average turnaround time experienced

Formulae for calculating turnaround time, waiting time and delay:

**Non pre-emptive:**

* **Average turnaround time experienced in non-pre-emptive scheduling is:**

Average turnaround time:

* **Average waiting time experienced in non-pre-emptive scheduling is:**

Average waiting time:

* **Average delay experienced in non-pre-emptive scheduling is:**

Average delay:

**Pre-emptive:**

* **Average turnaround time experienced in pre-emptive scheduling is:**

Average turnaround time:

* **Average waiting time experienced in pre-emptive scheduling is:**

Average waiting time:

* **Average delay experienced in pre-emptive scheduling is:**

Average delay:

## 3.3 Scheduling algorithm with better performance and its justification

**Bibliography**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Kinicki and Williams Irwin. (2008) *Management*, McGraw Hill.
2. Decenzo David and Robbin Stephen A. (1996) *Personnel and Human Reasons Management*, Prentice Hall of India.
3. J.A.F. Stoner, Freeman R. E and Daniel R Gilbert. (2004) *Management*, 6th Edition, Pearson Education.
4. Fraidoon Mazda. (2000) *Engineering Management*, Addison Wesley.

All referencing, bibliography needs to be done as described in the following article:

<http://www.msruas.ac.in/pdf_files/VCBlogs/Academic%20Good%20Practices.pdf>