**Tutorial 4**

**1. Construct a DFD for the following system.**

The department of public works for a large city has decided to develop a Web-based pothole tracking and repair system (PHTRS). A description follows: Citizens can log onto a website and report the location and severity of potholes. As potholes are reported, they are logged within a “public works department repair system” and are assigned an identifying number, stored by street address, size (on a scale of 1 to 10), location (middle, curb, etc.), district (determined from street address), and repair priority (determined from the size of the pothole).

Work order data are associated with each pothole and include pothole location and size, repair crew identifying number, number of people on crew, equipment assigned, hours applied to repair, hole status (work in progress, repaired, temporary repair, not repaired), amount of filler material used, and cost of repair (computed from hours applied, number of people, material and equipment used).

Finally, a damage file is created to hold information about reported damage due to the pothole and includes citizen’s name, address, phone number, type of damage, and dollar amount of damage. PHTRS is an online system; all queries are to be made interactively. Write down your assumptions clearly.

Define flow boundaries and map the DFD into a software architecture using transform analysis

2. **Develop a component level design for the following case study.**

Consider the module *ComputePageCost.* The intent of this module is to compute the printing cost per page based on specifications provided by  
the customer. Data required to perform this function are: number of pages in the document, total number of documents to be produced, one- or two-side printing, color requirements, and size requirements. These data are passed to *ComputePageCost* via the module’s interface. *ComputePageCost* uses these data to determine a page cost that is based on the size and complexity of the job- a function of all data passed to the module via  
the interface. Page cost is inversely proportional to the size of the job and directly proportional to the complexity of the job.

The *ComputePageCost* module accesses data by invoking the module *getJobData,* which allows all relevant data to be passed to the component, and a database interface, *accessCostsDB,* which enables the module to access a database that contains all printing costs.