You may be asked to demonstrate/explain your work to the tutor, if you are absent/unavailable or fail to demonstrate properly, zero marks will be awarded.

Text book: Deitel, H M & Deitel, P J 2013, C: How to program, 7th edn, Pearson Prentice-Hall, Upper Saddle River, New Jersey.

IMPORTANT: Submission Format

Copy and paste the question and then write your answer. If it is a programming question copy and paste your code from text editor followed by the screenshots of the output window. Marks will be deducted if this format is not followed. You need to follow the exact sequential number as in the tut sheet. Marks will be deducted if the submission format is not followed.

1. Write a statement or set of statements to accomplish each of the following. Assume that all the manipulations occur within the main function (therefore, no addresses of pointer variables are needed).
   Use the following structure definition to answer questions 2.a.b.c.d and e

```
struct bankEmployee {
      char name[20];
      int salary;
      struct bankEmployee *next;
};

typedef struct bankEmployee BANKEmployee;
typedef BANKEmployee *BANKEmployeePtr;
```

   a. Create a pointer to the start of the list called `startPtr`, the list is currently empty.

   b. Create a new node of type `BANKEmployee` that's pointed to by pointer `newPtr` of type `BANKEmployeePtr`. Assign `"Justin" as the name` and `1000 as the salary. Make startPtr` to point to this node. Provide any necessary declarations and statements.
   Use diagram to show the `startPtr and the new node.`

   c. Assume that the list pointed to by `startPtr` currently consists of 2 nodes one containing `"Justin"` and one containing `"Sam"`. Assume Sam's salary as 999 and the nodes are in alphabetical order.

   Use diagrams to show the insertion of the following nodes with these data for `name` and `salary`:

```
"Antony"        200
"Tony"          300
"Peter"         400
```

   Provide C programming statements to insert the above nodes

Use pointers `previousPtr`; `currentPtr` and `newPtr` to perform the insertions; State what `previousPtr` and `currentPtr` point to before each insertion. Assume that `newPtr` always points to the new node, and that the new node has already been assigned the data.

d. Write a while loop that prints the data in each node of the list. Use pointer `currentPtr` to move along the list.

e. Write a while loop that deletes all the nodes in the list and frees the memory associated with each node. Use pointer currentPtr and pointer tempPtr to walk along the list and free memory, respectively.

2. Create a linked list using the following structure

```
struct myID {

        int value;
        struct myID *next;
};

typedef struct myID MYID;
typedef MYID * MYIDPtr;
```

Use the above structure definition and create a linked list with 5 nodes. (No functions or loops to be used for the creation of the nodes).

Use the last five digits of your student id as the values for the 5 nodes of the linked list. The value of each node will contain one digit from your student id.

For example, if your student ID is 101285346, take the last five digits – 85346, so the insertion order should be as shown below.

```
newptr=……..malloc(MYID);
    newptr->value=8;
                5
                .
                .
                6
```

and the final linked list should be in ascending order of numbers as shown below.