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 studentname {
char letter;
struct studentname *next;
};

typedef struct studentname STUDENTName;
typedef STUDENTName *STUDENTNamePtr;
```

Create a linked list manually (without using any loops or recursive functions or any functions) that contains five nodes where the data part (structure element letter) of the nodes should be the first five letters of your last name. One letter will go to one node and the node insertion should happen one after another in alphabetical order. When you insert a new node, it should be in the right place to get alphabetical order as shown in previous task. If your last name doesn't have five letters fill the remaining nodes with letters from your first name starting from the first letter (Eg: if your name is Devin Ly, then nodes will contain L, y, D, e and v).

Example:

Assume your name is Ricky Ponting; take the first five letters of the last name, which is Ponti so the insertion order is

```
newptr = new STUDENTName;
newptr -> letter = 'P ';
                .
                .
                .
                o
                .
                .
                n
                .
                t
                .
                i
```

and when you print the linked list it should be like as shown below (Note, you are not using any sorting to get this in alphabetical order, you manually insert nodes in the right place to get it in alphabetical order).