II Semester

**Problem Solving with Data Structures**

# Introduction to Data Structures

2020-2021

# Motivation

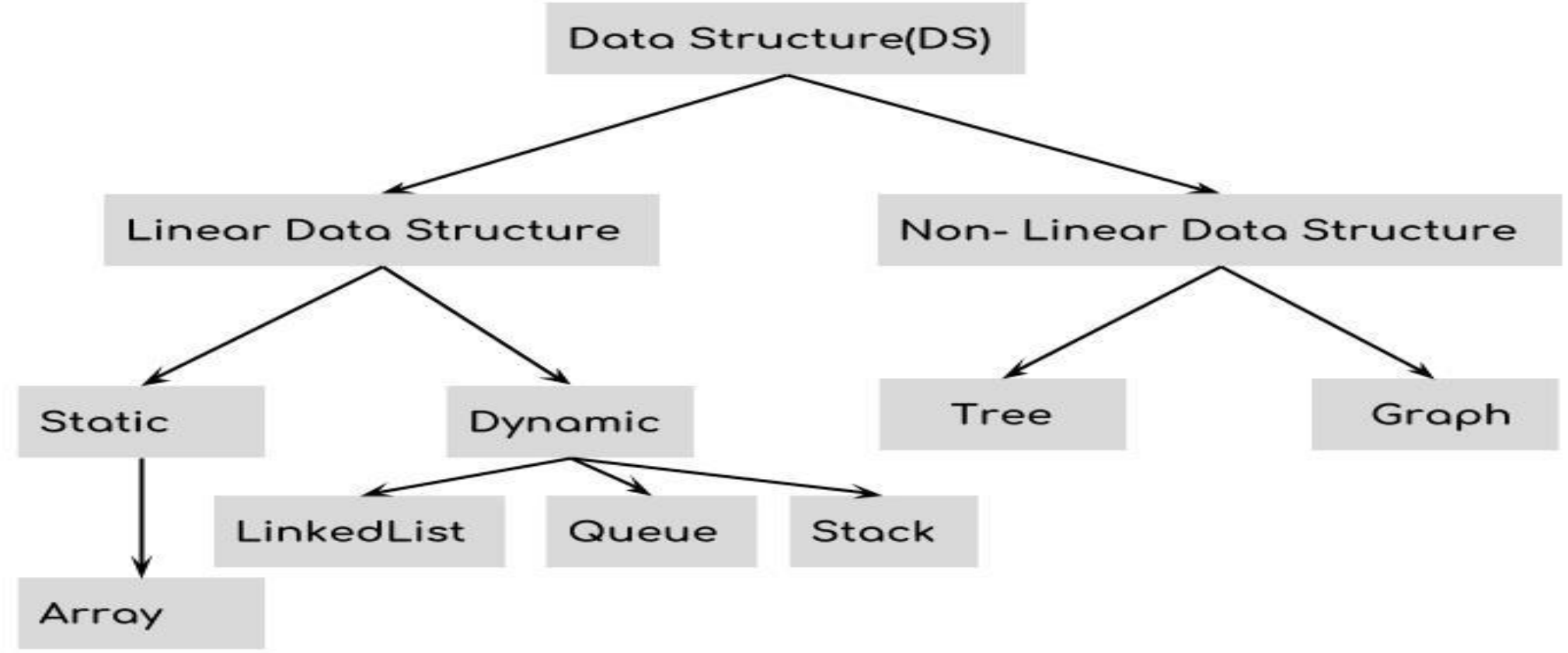
**A data structure is a special way of organizing and storing data in a computer so that it can be used efficiently.**

# Why we need data structures?

1. **Data Organization:** We need a proper way of organizing the data so that it can accessed efficiently when we need that particular data.

2. **Efficiency:** when we need to perform several operation such as add, delete update and search on arrays , it takes more time in arrays than some of the other data structures.

# Classification



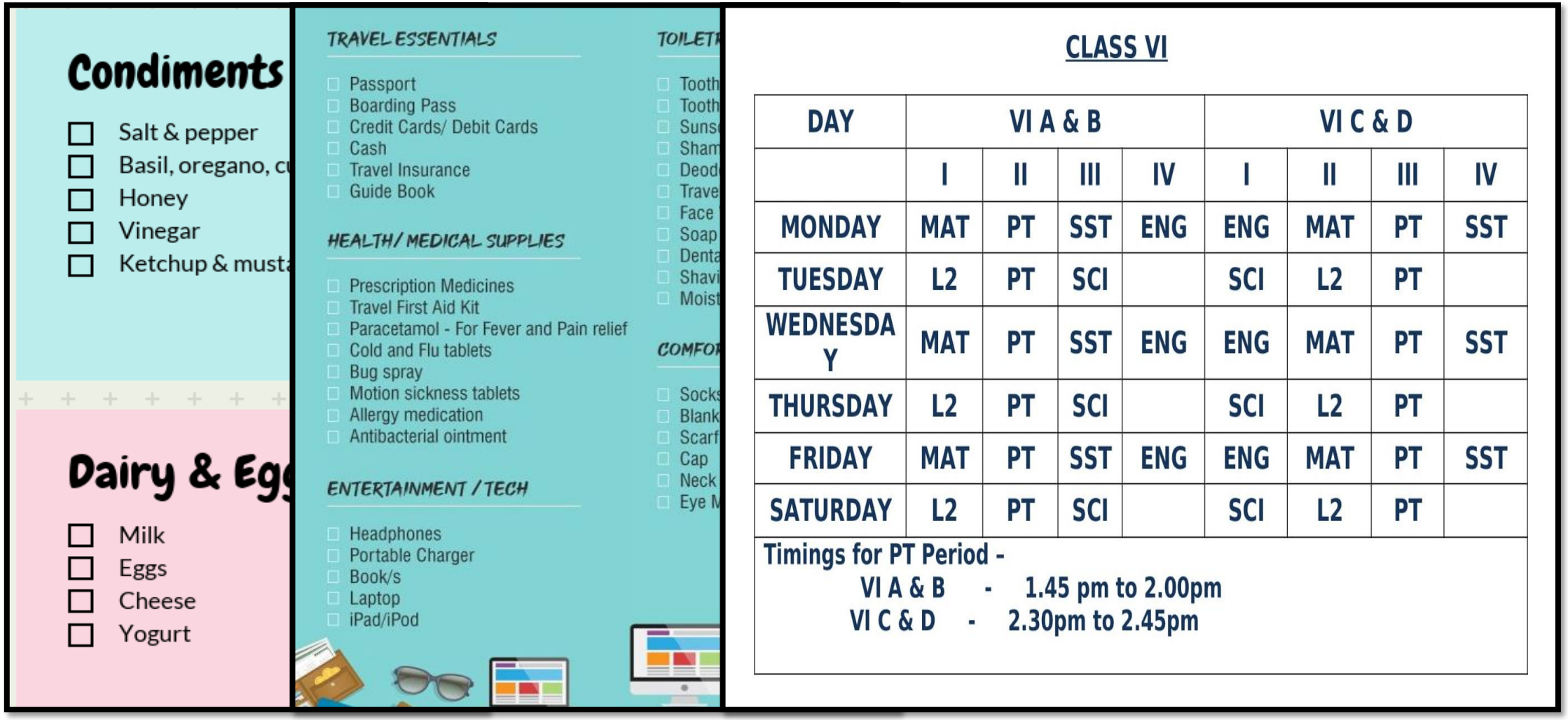
II Semester

**Problem Solving with Data Structures**

# List

2020-2021

# Motivation



## Condiments

- Salt & pepper
- Basil, oregano, cu
- Honey
- Vinegar
- Ketchup & musta

## Dairy & Egg

- Milk
- Eggs
- Cheese
- Yogurt

### TRAVEL ESSENTIALS

- Passport
- Boarding Pass
- Credit Cards/ Debit Cards
- Cash
- Travel Insurance
- Guide Book

### HEALTH/ MEDICAL SUPPLIES

- Prescription Medicines
- Travel First Aid Kit
- Paracetamol - For Fever and Pain relief
- Cold and Flu tablets
- Bug spray
- Motion sickness tablets
- Allergy medication
- Antibacterial ointment

### ENTERTAINMENT / TECH

- Headphones
- Portable Charger
- Book/s
- Laptop
- iPad/iPod

### TOILETR

- Tooth
- Tooth
- Sunsc
- Sham
- Deod
- Trave
- Face
- Soap
- Denta
- Shavi
- Moist

### COMFOR

- Socks
- Blank
- Scarf
- Cap
- Neck
- Eye M

## CLASS VI

| DAY | VI A & B | | | | VI C & D | | | |
|---|---|---|---|---|---|---|---|---|
| | I | II | III | IV | I | II | III | IV |
| MONDAY | MAT | PT | SST | ENG | ENG | MAT | PT | SST |
| TUESDAY | L2 | PT | SCI | | SCI | L2 | PT | |
| WEDNESDAY | MAT | PT | SST | ENG | ENG | MAT | PT | SST |
| THURSDAY | L2 | PT | SCI | | SCI | L2 | PT | |
| FRIDAY | MAT | PT | SST | ENG | ENG | MAT | PT | SST |
| SATURDAY | L2 | PT | SCI | | SCI | L2 | PT | |

Timings for PT Period -
VI A & B - 1.45 pm to 2.00pm
VI C & D - 2.30pm to 2.45pm

**GROCERY LIST** **TRAVELLING CHECK LIST** **CLASS TIME TABLE**

# List?

List is a sequence of "what to do" one by one.

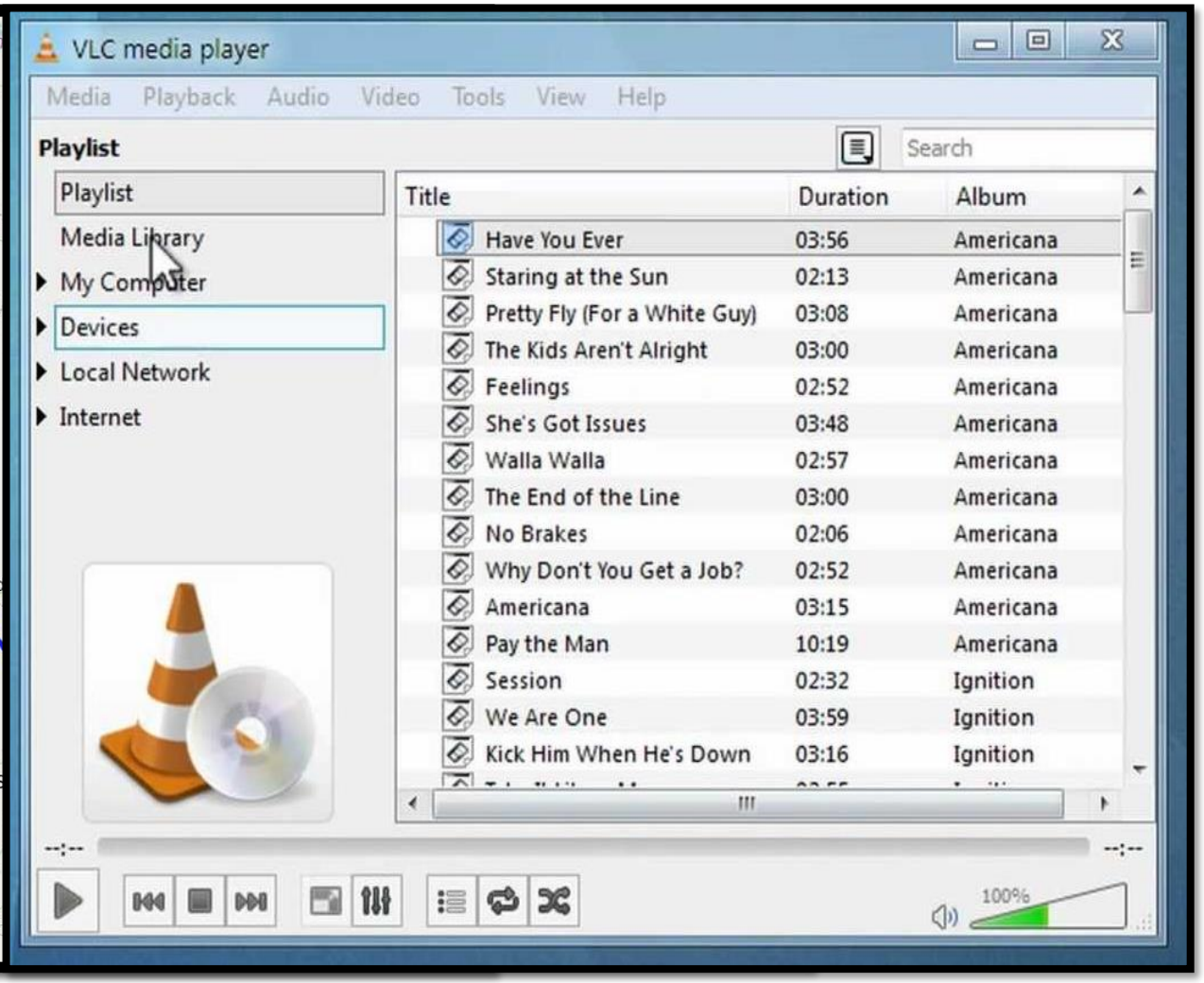
Class Activity – 1 (Individual)

1. Prepare a "TO-DO" list after returning home from college.
2. You are given 4 assignments to complete with different deadline dates. Prepare a Plan to complete the assignments based on deadline.

## Basic operations

1. Inserting: Insert a new data in the list
2. Deleting : To remove the existing data from the list
3. Iterating a list: Navigation of list
4. Searching : Find out a specific data in the list

# Motivation


**GAME OF TREASURE HUNT**

**ROUTE PLANNER**

**MEDIA PLAYER**

# List

Class Activity – 2 (Individual)

What is the difference between GROCERY LIST and GAME OF TREASURE HUNT?

# List

Can we store the linked data items in an array storage structure?

Answer: No.

Disadvantages of arrays

- Fixed in size.
- Insertions and Deletions are time consuming process.

How do we  solve the problem?

Answer: Linked list.

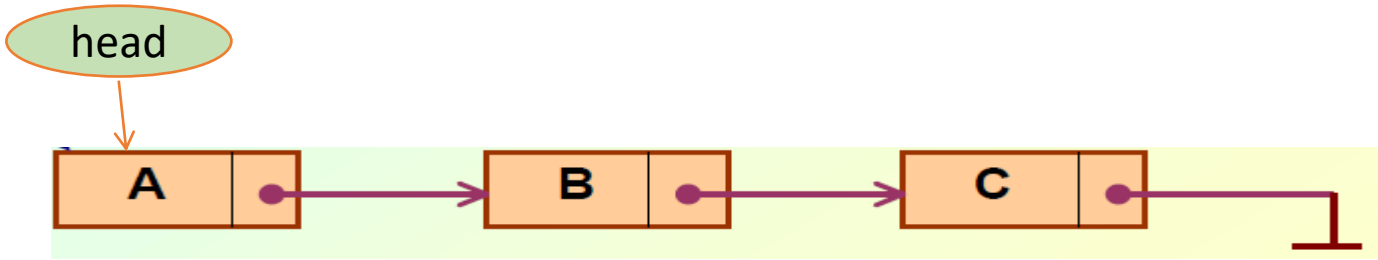
Linked list is a sequence of data items where each data item has the related information. Every data item is interconnected with each other.

# Linked List

A linked list is a data structure in which data items can be added and removed during run time of the program.

Characteristics of Linked List:

1. Dynamic behavior – Adding and removing of data items during runtime.
2. Successive elements are connected by pointers.
3. Last element points to NULL (for selective Linked Lists).
4. Keeping track of a linked list: know the pointer to the first element of the list (called as start, head, first etc…).

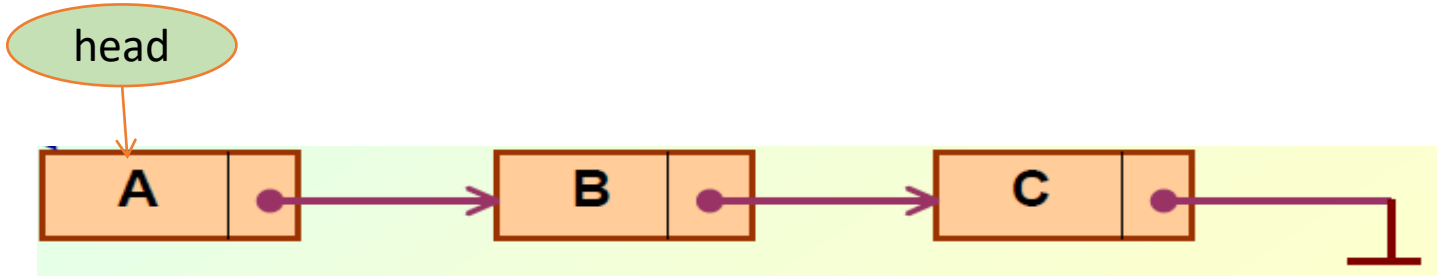


# Types

Depending on the way in which the links are used to maintain adjacency, several different types of linked lists are possible.

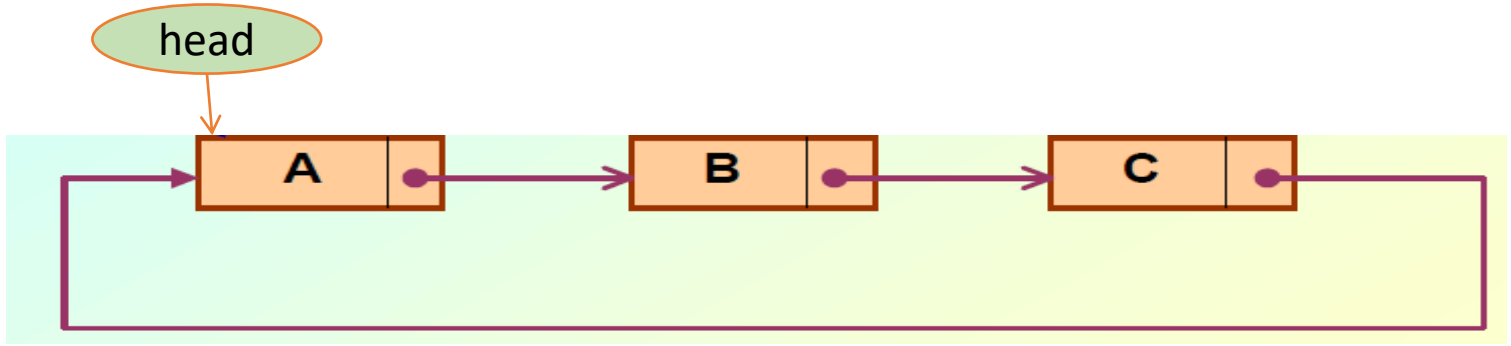
1. Singly-linked list
2. Circular Singly-linked list
3. Doubly linked list
4. Circular Doubly linked list

## Singly Linked List




## Circular Singly Linked List

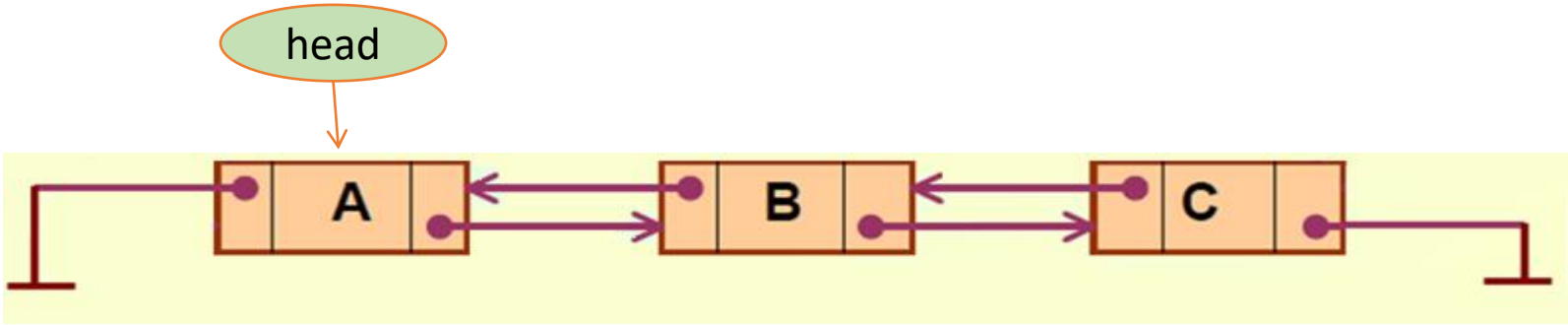
The pointer from the last element in the list points back to the first element.



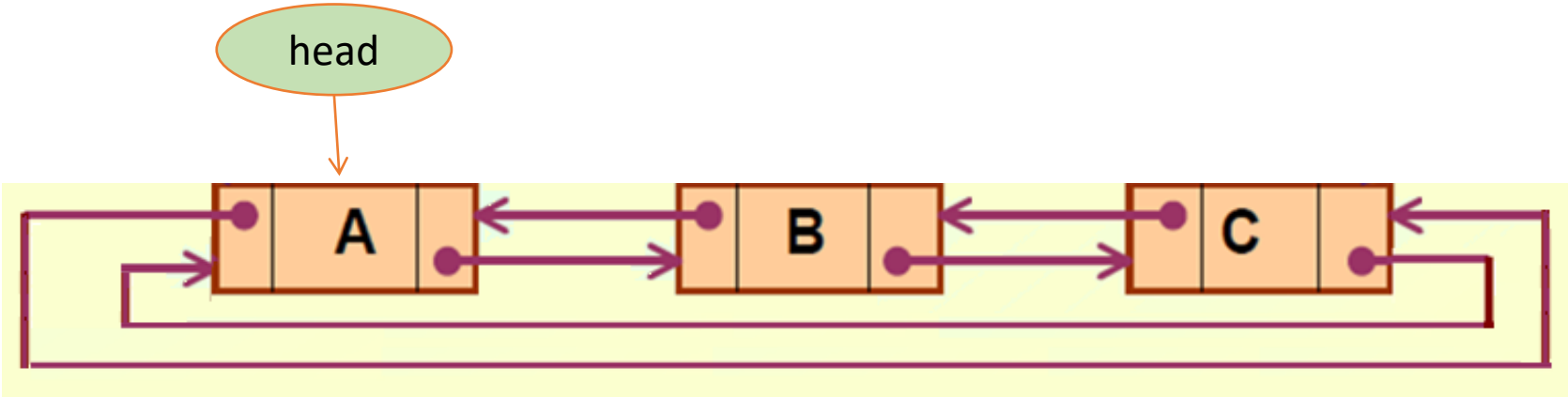
## Doubly Linked List

Pointers exist between adjacent nodes in both directions.

The list can be traversed either forward or backward.




## Circular Doubly Linked List



# Singly Linked List

# Singly Linked List – Basic Operations

1. Inserting an item in the Singly Linked List
   - At the End,
   - At the Front,
   - At the Specific position

2. Deleting an item from the Singly Linked List
   - From the End,
   - From the Front,
   - From the Specific position

3. Traversing the Singly Linked List

4. Searching the Singly Linked List

## Working with Singly Linked List

Class Activity – 3 (Individual)

Karnataka CET started admission for first year under graduate program for the year 2021. After the seat allotment process, candidates who got their seat allotted in ATU. started visiting the campus for admission. Every candidate provided his/her basic details for admission process. Management of ATU. wanted to know the admitted list of candidates after first round. Apply Problem Solving Framework to create and display the list of candidates.

# Problem Solving Framework

# Problem Solving Framework:

## I. Understanding the Problem:

1. Knowns: **Individual Candidate details.**
2. Unknowns: **List of Candidates admitted to ATU.**
3. Extra Information: **ATU. Admission Process**
4. Assumptions: **NIL**

## II. Devise a Plan:

1. Problem Representation: **Numerical**
2. Problem Solving Strategy: **Mathematical Reasoning**
3. Identification of Node and its Members:
   - Node: **Candidate.**
   - Members: **Name, Rank, Age, Address.**
4. Identification of Operations:
   - **Creation of Node.**
   - **Reading the data for each Node.**
   - **Insertion of a Node at the end of the List.**
   - **Displaying the List.**

## Problem Solving Framework:

### III. Carry Out a Plan:

Solution for the Plan in terms of Algorithm / modular C program.

### IV. Assess the Result:

Specify the Input and Output for the designed Algorithm / modular C program.

### V. Summary

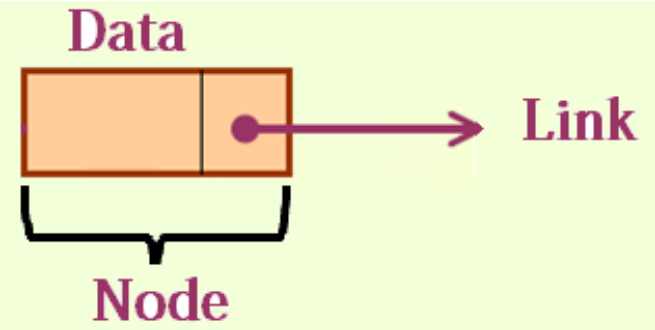
Summarize the way you have solved the problem.

# Node in the Linked List

**A Node:**

A structure which contains two important fields of information.

1. Data Part.
2. Link Part.



# Define a node for the Candidate

```
struct candidate
{
        char name[25];
        int rank, age;
        char address[100];
        struct candidate *next;
};
```

Pointer declared inside the structure, pointing to itself – **Self Referential Structure**

```
/* A user-defined data type called "NODE" */
typedef struct candidate *NODE;
```

typedef name given to self referential structure – **Pointer typedef name**

# Creating a Singly Linked List

# Creating a Singly Linked List

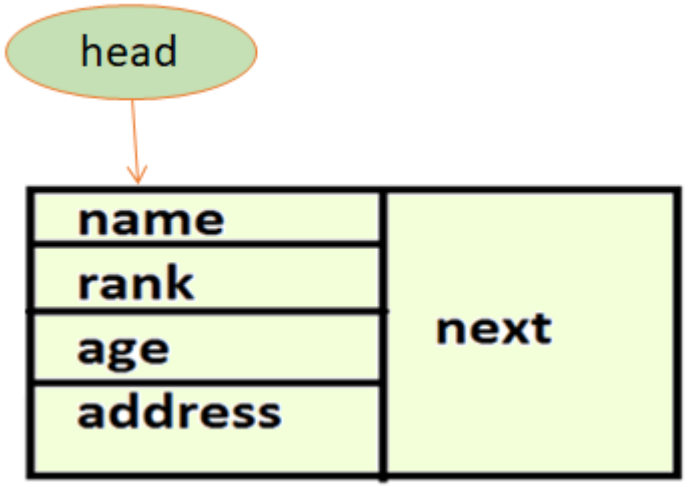
Perform Insert operation to create a Singly inked List.
We start with Insert at the End of the List.

**How to begin?**

To start with, we have to create a node (the first node), and make **head** point to it.

**Algorithm: To create a node**

Step 1: **Start**
Step 2: **Declare a pointer for node.**
Step 3: **Allocate the memory for node.**
Step 4: **Check whether memory is allocated.**
Step 5: **Return the address of node.**
Step 6: **Stop**



# Function to Create a node

```
NODE getnode()
{
   NODE new;
   new=(NODE)malloc(sizeof(struct candidate));
   if(new==NULL)
   {
          printf("Not created\n");
          exit(0);
   }
   new->next = NULL;
   return new;
}
```

# Insert Node at the End

# Inserting Nodes at the End

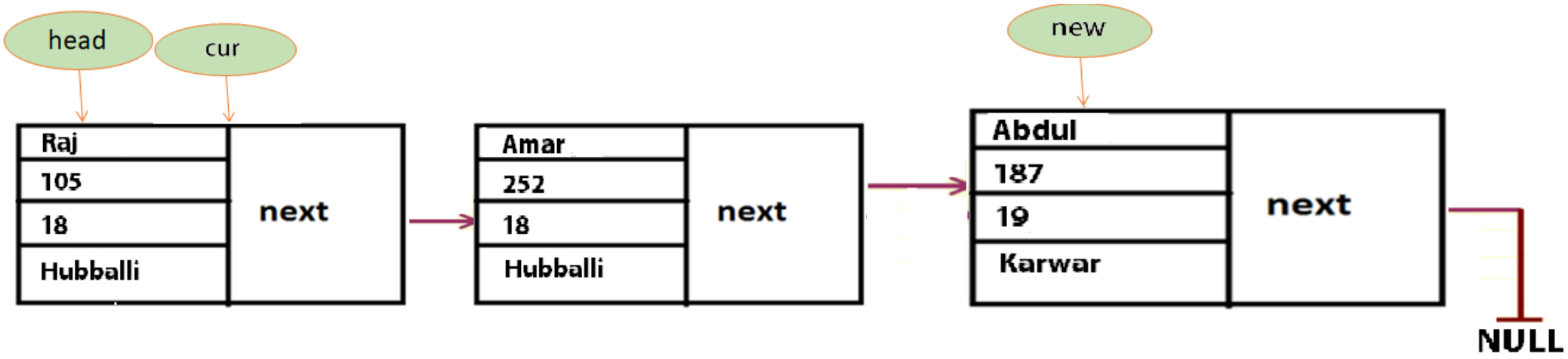
**To insert a node at the end of the List for more number of candidates:**

1. Create the nodes, one by one.
2. Read in the fields/members of the each node.
3. Modify the links of the nodes such that the **chain** is formed. To form the chain, we insert the nodes at the end of the List.

**Algorithm: To insert nodes at the end of the List.**

Step 1: **Start**
Step 2: **Declare a pointer for node.**
Step 3: **Create the node.**
Step 4: **Store [Read] the details.**
Step 5: **Check whether the list is empty.**
**if it is, then make the created node as head node.**
Step 6: **Traverse till end of list and then Connect node at the end.**
Step 7: **Return head node address [first node of the list].**
Step 8: **Stop.**

# Inserting Nodes at the End: Illustration



# Inserting Nodes at the End

To be called from main() function as:

```
NODE head;
…
…
…
head = insert_end (head);
```

# User Defined Function: Inserting Node at the End

```
NODE insert_end (NODE head)
{
        NODE new, cur;
        new = read_details();
        new ->next = NULL;
        if(head==NULL)
                return new;
        cur = head;
        while (cur->next != NULL)
        {
                cur = cur->next;
        }
        cur -> next = new;
        return head;
}
```

# Singly Linked List

## Read the details of each node

# To Read the details of each node/ Candidate

```
NODE read_details()
{
    NODE temp;
    temp = getnode();
    scanf("%s %d %d %s",temp->name,&temp->rank,&temp->age,temp->address);
    return temp;
}
```

# Insert Node at the Front

# Insert a node at the front of the List

To insert a node at the front of the List for more number of candidates:

**Algorithm: To insert a node at the front of List.**

Step 1: **Start**

Step 2: **Create a new node.**

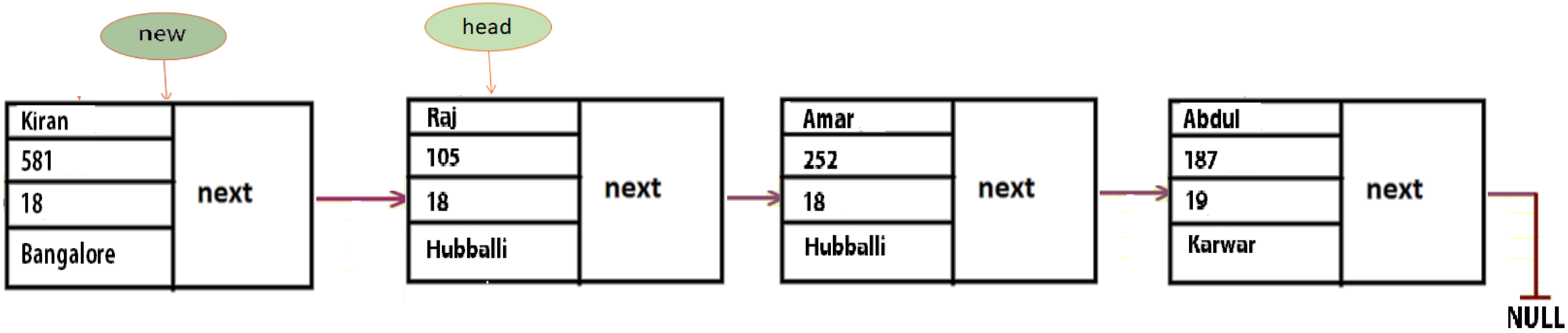
Step 3: **Store [Read] the details.**

Step 4: **Check whether the list is empty.**
**if it is, then link the newly created i.e. the new Node will now point to head node.**

Step 5: **Otherwise, Make the new node as the head node, i.e. now head node will point to new Node.**

Step 6: **Stop**

# Inserting Nodes at the Front: Illustration



# User Defined Function: Insert a node at the front

```
NODE insert_front( NODE head)
{
        NODE new;
        new = read_details();
        new -> next = NULL;
        if(head == NULL)
        {
                return new;
        }
        new ->next = head;
        head = new;
        return head;
}
```

## User Defined Function: Insert a node at the front

**To be called from main() function as:**

```
NODE head;
…
…
…
head = insert_front(head);
```

# Traversing and Displaying the List

# Traversing and Displaying the List

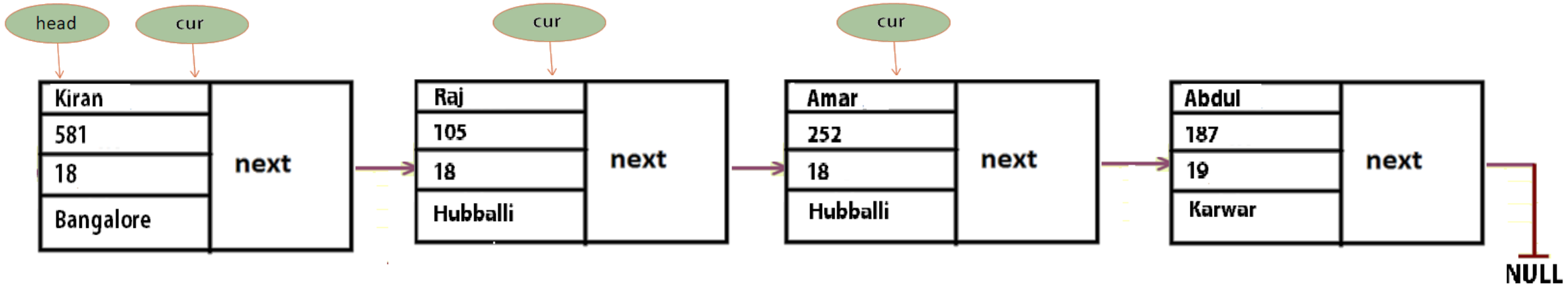
**What is to be done?**

Once the linked list has been constructed, head points to the first node of the list, start from head node and print the contents of each node on the screen.

**Algorithm: Display the list**

Step 1: **Start**
Step 2: **Check whether the list is empty.**
Step 3: **Follow the cur pointer.**
Step 4: **Display the contents of the every node as it is pointed by cur pointer**
Step 5: **Stop when the cur's next pointer reaches NULL.**
Step 6: **Stop**

# Traversing and Displaying the List




**OUTPUT**

| Name | Rank | Age | Address |
|---|---|---|---|
| Kiran | 581 | 18 | Bangalore |
| Raj | 105 | 18 | Hubballi |
| Amar | 252 | 18 | Hubballi |
| Abdul | 187 | 19 | Karwar |

# Display the List

```
void display_list (NODE head)
{
        NODE cur;
        if(head==NULL)
        {
           printf("Empty List\n");
           return NULL;
        }
        printf("elements are\n");
        cur = head;
        printf("Name\tRank\tAge\tAddress\n");
        while (cur != NULL)
        {
           printf("%s\t%d\t%t\t%s\n",cur->name,cur->rank,cur->age,cur->address);
           cur = cur ->next;
        }
}
```

# Display the List

**To be called from main() function as:**

```
NODE head;
…
…
…
display_list (head);
```

# Inserting at Specific Position

## Inserting at Specific Position

**Can be performed in different ways.**

1. Get the information of existing node from the list. Then connect the new node either:
   a. before the specified node.
   b. after the specified node.
2. Based on the node count, a new node can be connected

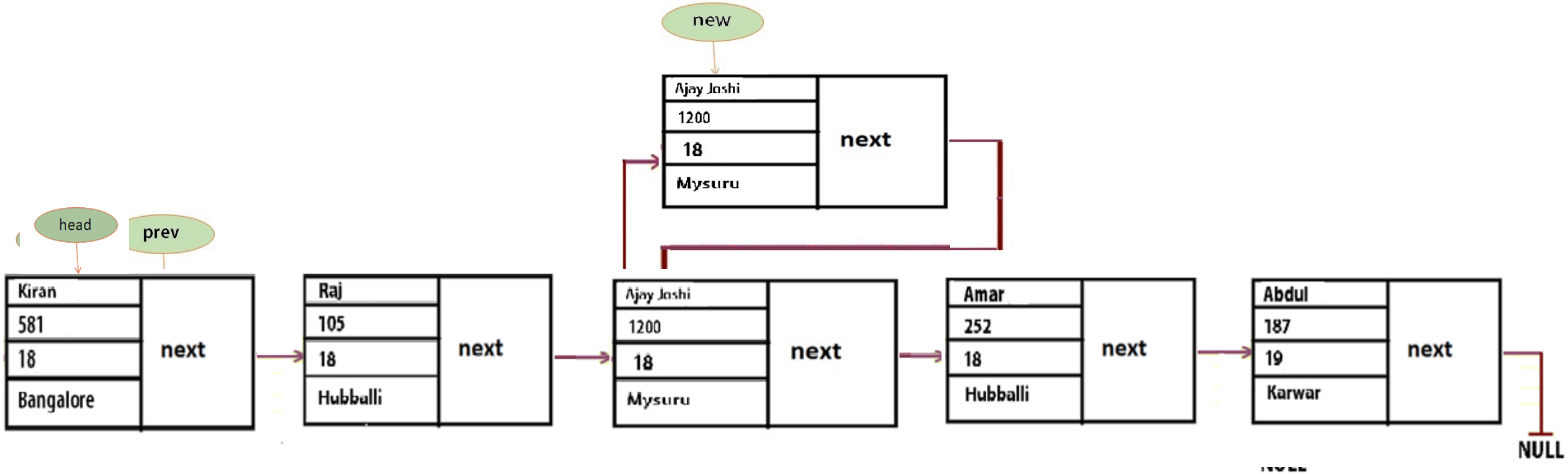
## Inserting before the Specified Node

**Algorithm: To insert before the specified node**

Step 1: **Start**
Step 2: **Create the new node.**
Step 3: **Store[Read] the data into new node.**
Step 4: **Read the existing node's data.**
Step 5: **Traverse using two pointers prev and cur, till the existing node. prev pointer should follow the cur pointer [i.e. prev pointer will be one node behind the cur pointer].**
Step 6: **Connect prev node's next to new node.**
Step 7: **Connect new node's next to cur node.**
Step 8: **Return the address of head node.**
Step 9: **Stop**

# Inserting before the Specified Node

**Insert before the existing node: Amar**



# Singly Linked List

**CLASS ACTIVITY – 4**

Karnataka CET started admission for first year under graduate program for the year 2021. After the seat allotment process, candidates who got their seat allotted in ATU. started visiting the campus for admission. Every candidate provided the his/her basic details for admission process. Management of ATU wanted to know the admitted list of candidates after first round. Apply Problem Solving Framework to create and display the list of candidates.

During the second round of admission process, the last candidate from the list who is from Mangalore, got admission in the Engineering College in his hometown. He withdrew his admission from ATU. Display the updated list of candidates admitted in ATU after withdrawal of last candidate.

# Delete node from the list

# Delete a node from the list

**To delete a node from the Singly Linked list:**

1. Delete a node from the End.
2. Delete a node from the Front.
3. Delete a node from the specific position.

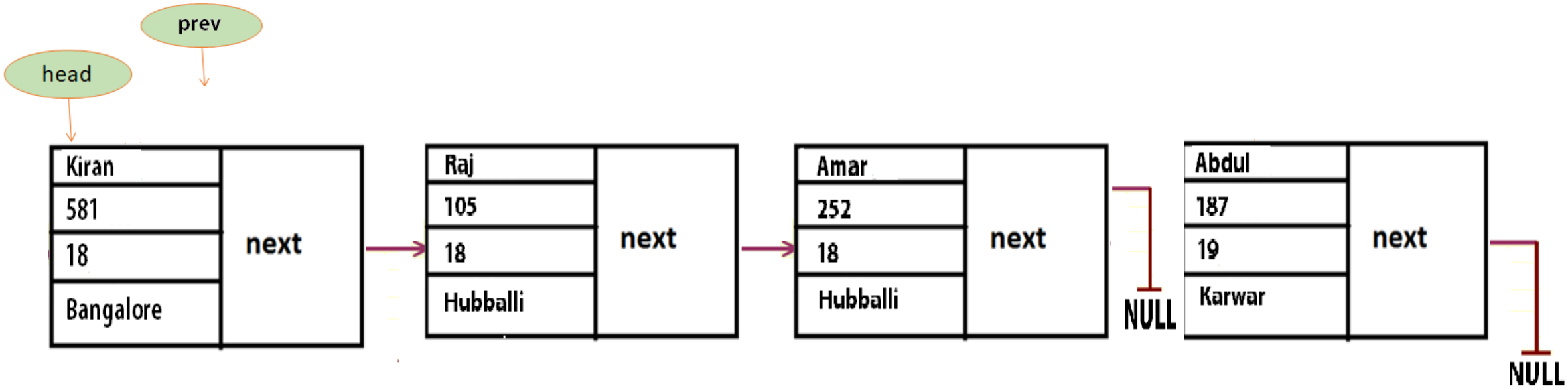
# Delete a node from End

# Delete a node from End of the list

**Algorithm: To delete a node from end of the List.**

Step 1: **Start**
Step 2: **Check whether the List is empty.**
**if yes, display message that List is empty.**
Step 3: **Check whether the List has only one node.**
**if yes, display the data and make head=NULL.**
Step 4: **Otherwise, Declare two pointers, prev and cur.**
Step 5: **Assign address of head node to cur.**
Step 6: **Traverse till the last node of the List. Pointer prev points to last but one node. Pointer cur points to last node.**
Step 7: **Disconnect the prev from cur [last node].**
Step 8: **Display the data of cur [last node].**
Step 9: **Deallocate the memory of cur [last node].**
Step 10**: Stop**

# Delete a node from End of the list



# Delete a node from End of the list

```
NODE delete_end(NODE head)
{
    NODE prev, cur;
    if(head==NULL)
    {
        printf("List Empty\n");
        return head;
    }
    if(head->next==NULL)
    {
      printf("Deleted: %s\n", head->name);
        free(head);
        return NULL;
    }
    prev =NULL;
     cur =head;
    while(cur ->next != NULL)
    {
        prev= cur;
         cur = cur ->next;
    }
    printf("Deleted: %s\n", cur->name);
    free(cur);
    prev->next=NULL;
    return head;
}
```

## CLASS ACTIVITY – 5

Karnataka CET started admission for first year under graduate program for the year 2021. After the seat allotment process, candidates who got their seat allotted in ATU started visiting the campus for admission. Every candidate provided the his/her basic details for admission process. Management of ATU. wanted to know the admitted list of candidates after first round. Apply Problem Solving Framework to create and display the list of candidates.

During the second round of admission process, the last candidate from the list who is from Mangalore, got admission in the Engineering College in his hometown. He withdrew his admission from ATU. Display the updated list of candidates admitted in ATU, after withdrawal of last candidate.

During the third round of Admission process, the first candidate in the list obtained the admission in IIT Dharwad, he withdrew his admission from ATU. Display the updated list of candidates admitted in ATU.

# Delete node from Front

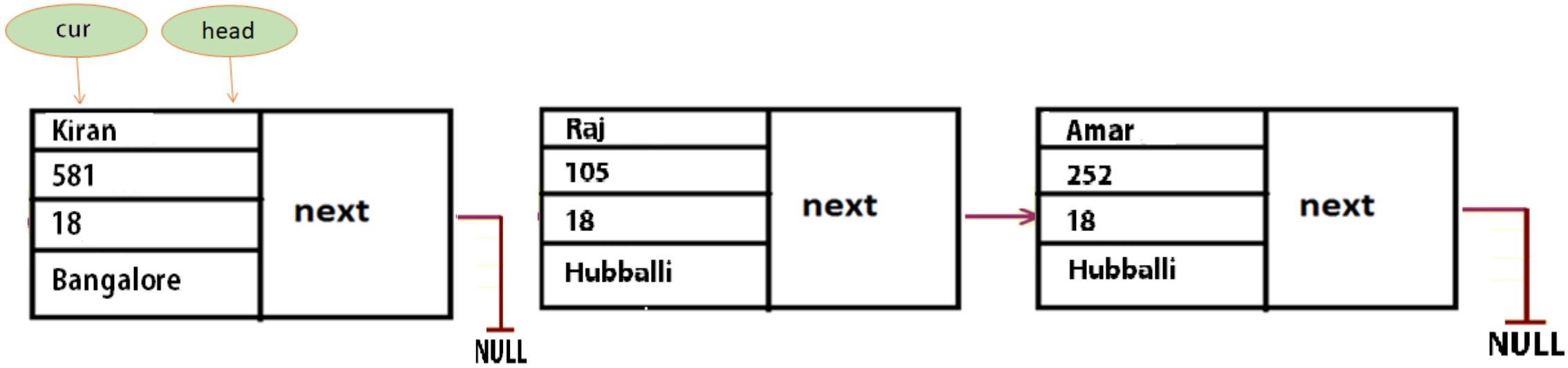
# Delete a node from Front of the list

**Algorithm: To delete a node from front of List.**

Step 1: **Start**
Step 2: **Check whether the List is empty.**
**if yes, display message that List is empty.**
Step 3: **Otherwise, Declare a pointer, cur.**
Step 4: **Assign address of head node to cur.**
Step 5: **Move head pointer to next node.**
Step 6: **Disconnect the cur [first node] from the list.**
Step 7: **Display the data of cur [first node].**
Step 8: **Deallocate the memory of cur [first node].**
Step 9: **Stop**

After the above operation, the **second node** in the Singly linked list becomes the head node of the list.

# Delete a node from Front of the list



# User Defined Function: Delete a node from Front

```
NODE delete_front (NODE head)
{
    NODE cur;
    if(head==NULL)
    {
        printf("List Empty\n");
        return head;
    }
    cur=head;
    head=head->next;
    printf("Deleted: %s\n", cur->name);
    free(cur);
    return head;
}
```

# Delete node from Specific Position

# Delete a node from Specific Position

**To delete the specified node: read the details of node to delete.**

**Algorithm: To delete a specific node.**

Step 1: **Start**

Step 2: **Read existing node's information to delete.**

Step 3: **Traverse till the specified node using two pointers prev and cur. Prev will be pointing to one node before the node to be deleted.**

Step 4: **Connect prev node's next to cur node's next node.**

Step 5: **Display the cur node's information.**

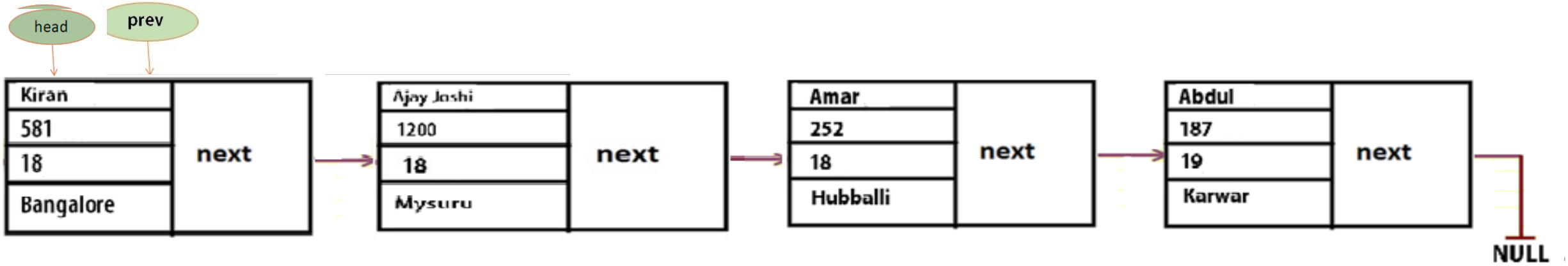
Step 6: **Deallocate the cur node's memory.**

Step 7: **Return the address of head node.**

Step 8: **Stop.**

# Delete a node from Specific Position

**Delete the node with name: Raj**



# PRACTICE PROBLEMS

1. Chicago college Placement Officer Mr. Mahon Raj has all the details of the students who are placed from the college along with the company name and passed out year. Since the ABN visit is in March 2021, Placement Officer has two tasks. The first one displays the students who are placed in 2020 and the second one displays the student details who has the highest Package. Please help Mr. Mahon Raj to do the tasks.

# TAKE HOME TASK

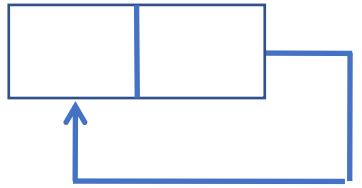
1. A development team in TCS plans a trekking trip to Kodachadri. The trip admin keeps track of people joining the trip. After 2 days few people drop out and 3 new people gets added. Help the trip admin to check the total number of people ready for the trip and create a final list.

# Circular Singly Linked List

# Introduction

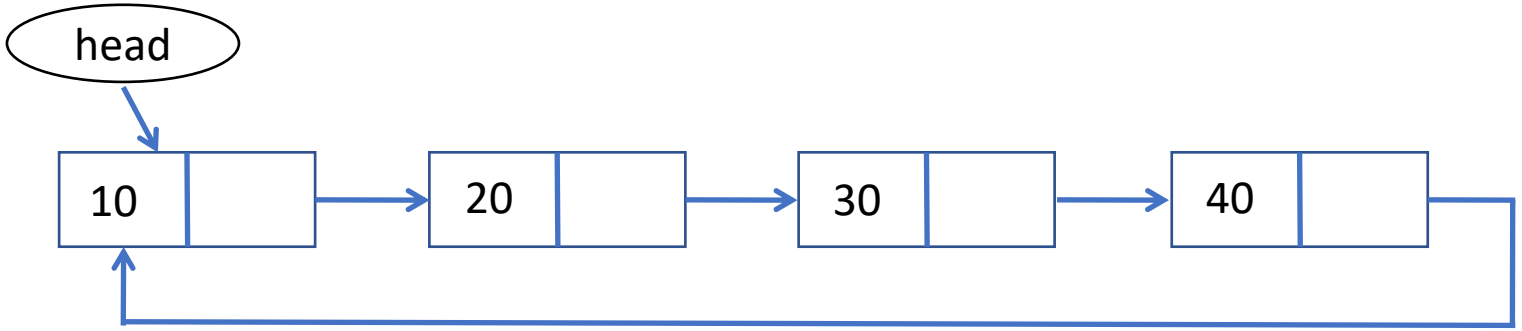
Circular linked list is a linked list where all nodes are connected to form a circle.

There is no NULL at the end.



Single Node of Circular List

We can traverse the whole list by starting from any node. We just need to stop when the first visited node is visited again.



# Introduction

**Class Activity-1 (Individual)**

**Everyday a Postman starts distribution of the letters at 10.00 am in Lingaraj Nagar area and returns to ABB CAMPUS post office at 12.00 noon. On first Day of the week, postman has 3 letters to deliver in his allotted area. He prepares a list of visiting houses before leaving the Post Office and notes down the delivery address of each house. Apply Problem Solving Framework to create and display the list of houses visited by the postman.**

# Problem Solving Framework

# Problem Solving Framework

**I. Understanding the Problem:**

1. Knowns: **letters to deliver.**
2. Unknowns: **List of houses to visit based on delivery address on letters.**
3. Extra Information: **Lingaraj nagar area , ABB CAMPUS post office.**
4. Assumptions: **Creation of list of letters on a particular day.**

**II. Devise a Plan:**

1. Problem Representation: **Numerical**
2. Problem Solving Strategy: **Mathematical Reasoning**
3. Identification of Node and its Members:
   - Node: **Letter.**
   - Members: **Name, House No, Area, mobile no.**
4. Identification of Operations:
   - **Creation of Node.**
   - **Reading the data for each Node.**
   - **Insertion of a Node in a List.**
   - **Displaying the List.**

# Problem Solving Framework

**III. Carry Out a Plan:**

Solution for the Plan in terms of Algorithm / modular C program.

**IV. Assess the Result:**

Specify the Input and Output for the designed Algorithm / modular C program.

**V. Summary**

Summarize the way you have solved the problem.

# Define a node for the letter

```
struct letter
{
        char name[25];
        int hno;
        char area [100];
         long int mobileno;
        struct letter *next;
};
```

Pointer declared inside the structure, pointing to itself – **Self Referential Structure**

```
/* A user-defined data type called "NODE" */
typedef struct letter *NODE;
```

typedef name given to self referential structure – **Pointer typedef name**

# Create a node

**Algorithm: To create a node in Circular List**

Step 1: **Start**

Step 2: **Declare a pointer for node.**

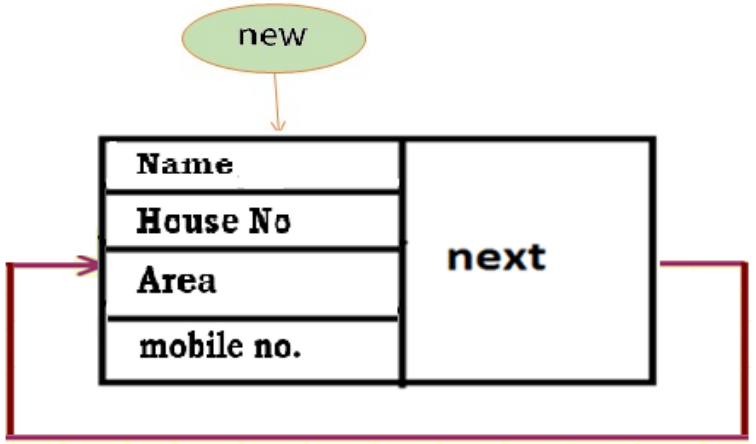
Step 3: **Allocate the memory for node.**

Step 4: **Check whether memory is allocated.**

Step 5: **Make the next pointer point to same node (circular!).**

Step 5: **Return the address of node.**

Step 6: **Stop**



# Inserting a Node in the Circular List

# Inserting a Node in Circular Singly Linked List

**Can be performed in following ways:**

- At the End,
- At the Front,
- At the Specific position.

# Inserting a Node at the End

# Inserting a Node at the End

**Algorithm: To insert at the end of Circular list**

Step 1: **Start**

Step 2: **Create the new node.**

Step 3: **Set the new node's next to itself (circular!).**

Step 4: **If the list is empty, return new node.**

Step 5: **Otherwise, Traverse till last node of the list [using cur pointer].**

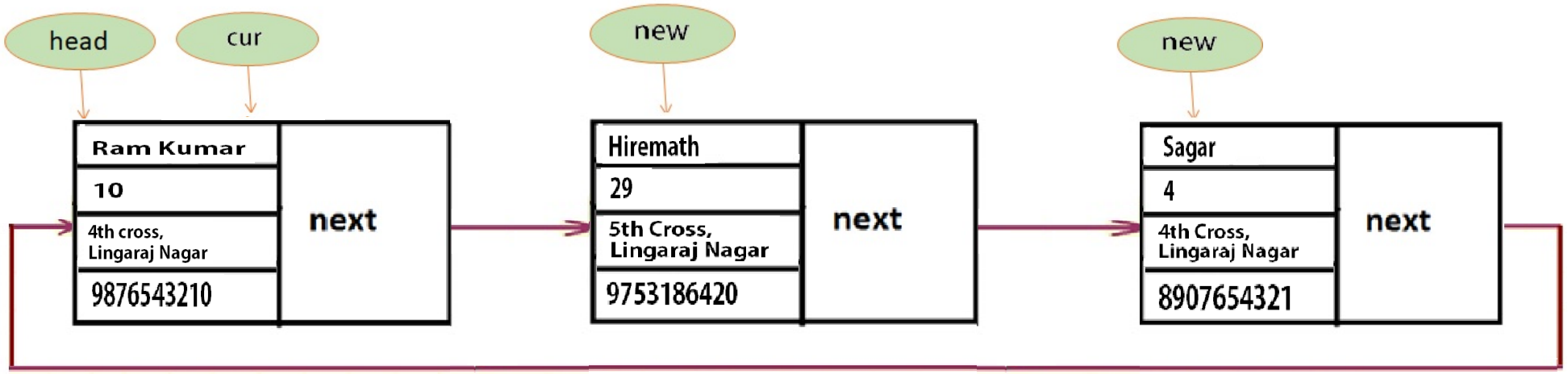
Step 6: **Connect cur node's next [last node] to new node.**

Step 7: **Connect new node's next to first node.**

Step 8: **Return the address of first node of the list.**

Step 9: **Stop**

# Inserting a Node at the End



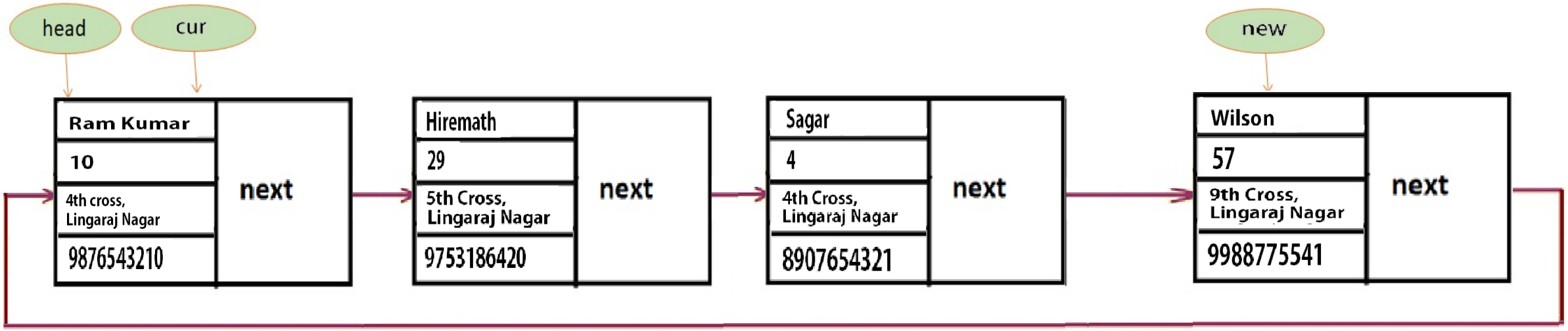
## Class Activity - 2 (Individual)

**Everyday a Postman starts distribution of the letters at 10.00 am in Lingaraj Nagar area and returns to ABB CAMPUS post office at 12.00 noon. On first Day of the week, postman has 3 letters to deliver in his allotted area. He prepares a list of visiting houses before leaving the Post Office and notes down the delivery address of each house. Apply Problem Solving Framework to create and display the list of houses visited by the postman.**

**Consider second Day of the week, postman has 1 more additional letter to be delivered compared to first Day (4 letters) to distribute in his allotted area. The $4^{th}$ letter belongs to the house which is located at the end of all the houses which he visited on first Day. Update and Display the list of houses to be visited by the postman on second Day.**

# Inserting a Node at the End

**After 3 nodes are inserted, the Circular List looks like:**



## Inserting a Node at the End

```
NODE insert_end (NODE head)
{
        NODE new, cur=head;
        new = read_details( );
        if(head == NULL)
        {
                new->next=new;
                return new;
        }
        else
        {
                while(cur -> next != head)
                        cur =cur -> next;
        }
        cur -> next = new;
        new -> next = head;
        return head;
}
```

# Read the details of LETTER node

```
NODE read_details()
{
    NODE temp;
    temp=getnode();
    scanf("%s%d%s%ld",temp->name, &temp->hno, temp->area, &temp->mobileno);
    return temp;
}
```

## Class Activity - 3 (Individual)

**Everyday a Postman starts distribution of the letters at 10.00 am in Lingaraj Nagar area and returns to ABB CAMPUS post office at 12.00 noon. On first Day of the week, postman has 3 letters to deliver in his allotted area. He prepares a list of visiting houses before leaving the Post Office and notes down the delivery address of each house. Apply Problem Solving Framework to create and display the list of houses visited by the postman.**

**Consider second Day of the week, postman has 1 more additional letter to be delivered compared to first Day (4 letters) to distribute in his allotted area. The 4th letter belongs to the house which is located at the end of all the houses which he visited on first Day. Update and Display the list of houses to be visited by the postman on second Day.**

**Consider Day 3 of the week, postman has 1 more additional letter compared to first and second Day (total 5 letters) to distribute in his allotted area. The letter belongs to the house which is located very near to his post office. Postman plans to deliver letter to nearer house first. Update and display the list of houses to be visited by the postman on Day 3.**

# Inserting a Node at the Front

# Inserting a Node at the Front

**Algorithm: To insert at front of Circular List**

Step 1: **Start**

Step 2: **Create the new node.**

Step 3: **Set the new node's next to itself (circular!).**

Step 4: **If the list is empty, return new node.**

Step 5: **Otherwise, Set new node's next to the first node [head node].**

Step 6: **Traverse to last node using cur pointer.**

Step 7: **Connect new node's next to first node.**

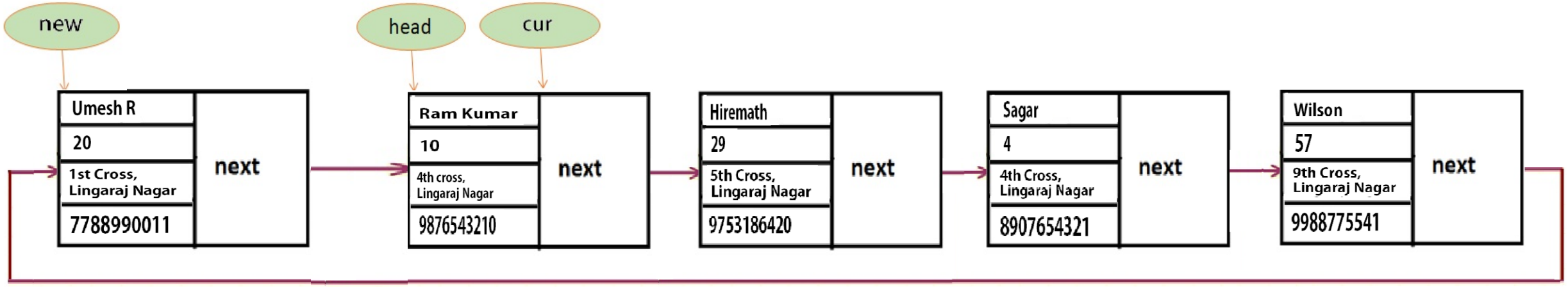
Step 8: **Connect cur node's next [last node] to first node.**

Step 9: **Shift the first node pointer to new node. New node becomes first node.**

Step 10: **Return the first node's address of the list.**

Step 11: **Stop**

# Inserting at the Front



# User Defined Function: Insert at the Front

```
NODE insert_front (NODE head)
{
        NODE new, cur=head;
        new = read_details();
        if (head ==NULL)
        {
                head = new;
                new -> next = head;
                return head;
        }
        while(cur ->next != head)
        {
                cur = cur -> next;
        }
        cur -> next=new;
        new->next=head;
        head = new;
        return head;
}
```

**Class Activity – 4 (Individual)**

**Postman contains a list of houses where he has to deliver the letters. During his second week of work, on Day 1 he has 4 letters to be delivered to the same houses to those he had delivered last week except the last house in the list. Now he has to remove the last house details from the list. Update and display the list by removing details of last house from the List.**

# Delete a node from Circular List

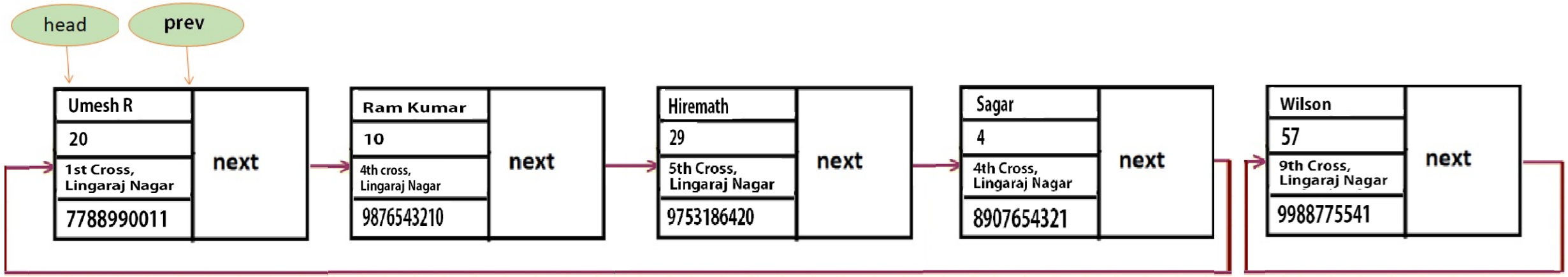
# Delete a node from Circular List

**To delete a node from the Circular Singly Linked list:**

1. Delete a node from the end.
2. Delete a node from the front.
3. Delete a node from the specified position.

# Delete a node from the End

# Delete a node from End



# User Defined Function: Delete a node from the End

```
NODE delete_end (NODE head)
{
    NODE prev=NULL, cur=head;
    // CASE 1: When list is empty
    if(head ==NULL)
    {
          printf("List empty\n");
          return NULL;
    }
  //CASE 2: When list contains only one node
    if(head->next==first)
    {
       printf("House deleted: %d\n", head->hno);
       free(head);
       return NULL;
    }
```

```
// CASE 3: When multiple nodes present
    cur = head;
    while(cur -> next != first)
    {
          prev=cur;
          cur=cur->next;
    }
    prev->next=head;
    printf("House deleted: %d\n",cur->hno);
    free(cur);
    return head;
}
```

**Class Activity - 5 (Individual)**

**Postman contains a list of houses where he has to deliver the letters. During his second week of work, on Day 2 he has 4 letters to be delivered to the same houses to those he had delivered last week, except the first house in the list. Update and display the list by removing details of first house from the List.**

# Delete a node from the Front

# Delete a node from the Front

**Algorithm: To Delete node from begin of Circular List**

Step 1: **Start**

Step 2: **Check whether list is Empty**

Step 3: **Check whether list is having only one node.**

**Delete the list and make the head pointer NULL.**

Step 4: **If the list contains more than one nodes:**

Step 5: **Traverse the list using single pointer to reach to last node of the list.**

Step 6: **Mark a new pointer to first node.**

Step 7: **Display the details of first node.**

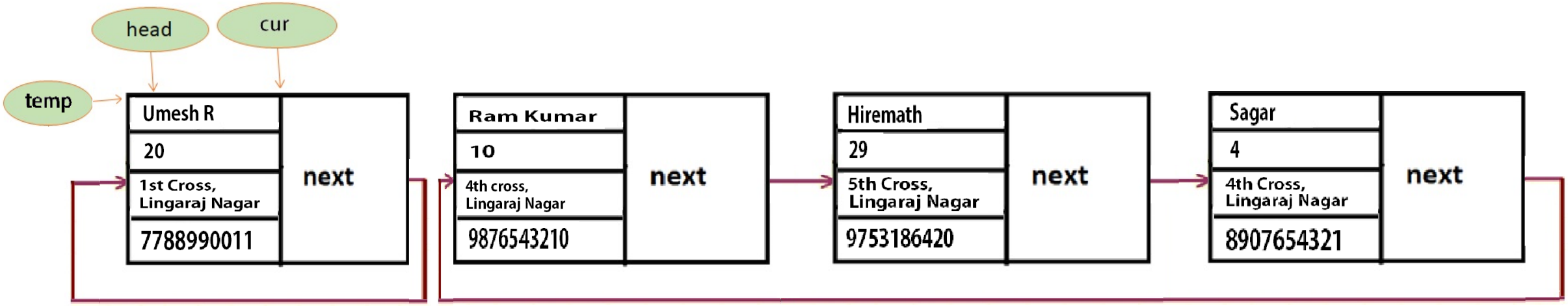
Step 8: **Shift the head pointer to second node of the list.**

Step 9: **Deallocate the memory of node pointed by new pointer.**

Step 10: **Connect the last node's next to second node (which now becomes first node).**

Step 11: **Stop**

# Delete a node from the Front



# Delete a node from the Front.

```
NODE delete_front (NODE head)
{
    NODE temp, cur=head;
    if(head==NULL)
    {
        printf("List empty\n");
        return NULL;
    }
    if(head->next==head)  //single node
    {
        printf("Deleted house: %d\n",head->hno);
        free(head);
        return NULL;
    }
        while(cur->next != head)
        {
                cur=cur->next;
        }
        temp=head;
        head=temp->next;
        cur->next=head;
        printf("Deleted house: %d\n",temp->hno);
        free(temp);
        return head;
}
```

# PRACTICE PROBLEMS

1. Wilson is resident of India, returns from America to India for a short visit. He wishes to go for Brand Factory for shopping. Wilson buys 3 jeans of different companies each with 40%, 50% and 60% discount and 3 casual shirts of different companies with 40% discount on each. Apply Problem Solving to Prepare a bill for Wilson using Circular Singly Linked List which stores the details of each item purchased. Calculate Total Bill amount and Display the bill.

# TAKE HOME TASK

1. After getting her PhD, Christie has become a celebrity at her university, and her facebook profile is full of friend requests. Being the nice girl she is, Christie has accepted all the requests. Now Kuldeep is jealous of all the attention she is getting from other guys, he asks her to delete some of the guys from her friend list. To avoid a 'scene', Christie decides to remove some friends from her friend list, since she knows the popularity of each of the friends she has, she removes some friends who are less popular than others. Help Christie to retain friends who are more popular and Kuldeep still being a good friend.