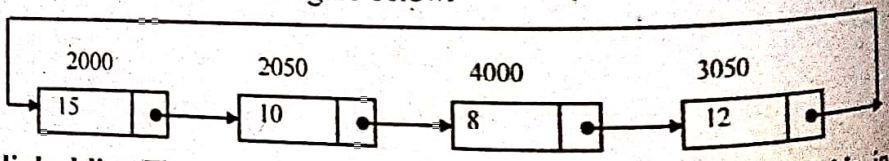
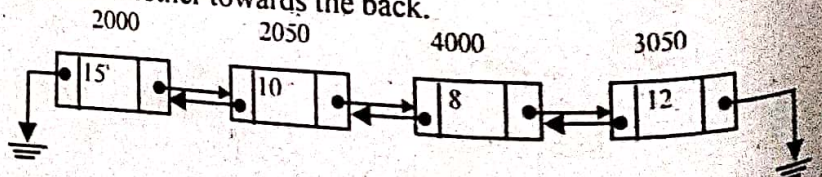
LINKED LIST

Chapter at a Glance

- **Singly linked lists:** A linked list is a data structure where data can be represented as a c of nodes. Each node of a linked list contains two parts: one is the address part another is data part. The address part holds the address of the node which is next to or previous to current node.
Typically the first node of a linked list is called the HEAD node. If the head node is destro then the entire list gets destroyed as well. Depending on the traversal requirement a linked can be designed as circular, bi-directional etc. In it's simplest form the following diag shows the structure of a uni-directional linked list also known as singly linked list.
- **Several operations:** (Insertion, deletion, Traversing, Searching)
- **Insertion:** The algorithm for inserting in the above manner is given below:
Let us assume that x is data value to be inserted after the node containing the data value initially contains the address of the head node.
  Step 1: start traversing the linked list from the head node
  Step 2: if the data value of the head node is not equal to y goto step 8
  Step 3: create a new node
  Step 4: place x to the data field of the new node
  Step 5: place the next address of the head node to the next address of the new node
  Step 6: change the next address of the head node by the new node address
  Step 7: Goto Step 13
  Step 8: Identify the address of the node containing data value y as follows
  Step 8a) if the data value of the next node of p is equal to y
  Step 8b) else change p by its next node address
  Step 9: create a new node
  Step 10: place y to the data field of the new node
  Step 11: place the next address of p node to the next address of new node
  Step 12: change the next address of the p node by the new node address
  Step 13: End
- **Circular linked list:** the circular linked list, the address of the last node contains the a of the first node, as shown in the figure below:



- **Doubly linked list:** The node of a doubly linked list contains two link fields, instead of One link is used to point to the predecessor node, i.e., the previous node & the other used to point to the successor node, i.e., the next node. So, the two links are direct towards the front and another towards the back.



- **Linked representa**

Each a_i x^i is i^{th} ter exponent. If a_i=0, th

1. In a circularly l modification of
a) no pointer
Answer: (c)

2. Linked list are not
a) Stack
Answer: (d)

3. Dynamic memory
a) Calloc
Answer: (d)

4. Which type of link in the sequence?
a) Singly Linked
c) Doubly Linke
Answer: (c)

5. Linked list is n problems?
a) insertion sort
c) binary search
Answer: (c)

6. Self-referential po
a) an array
c) a queue
Answer: (d)

7. Inserting a node a
a) four pointer e
c) one pointer e
Answer: (b)

8. Binary search is n
a) array
Answer: (b)

Linked representation of polynomial: In general any polynomial A(x) can be written as

$$A(x) = a_0 + a_1x + a_2x^2 + \text{-----} + a_{n-1}x^{n-1} + a_nx^n.$$

Each $a_i x^i$ is i^{th} term of the polynomial, where x is variable, a_i is its coefficient & e is the exponent. If $a_i=0$, then the term is zero term, otherwise it is nonzero term.

Multiple Choice Type Questions

1. In a circularly linked list organization, insertion of a record involves the modification of [WBUT 2008, 2018]
   a) no pointer b) 1 pointer c) 2 pointers d) 3 pointers
   Answer: (c)

2. Linked list are not suitable for [WBUT 2012, 2016]
   a) Stack b) Deque c) AVL Tree d) Binary search
   Answer: (d)

3. Dynamic memory allocation use [WBUT 2012]
   a) Calloc b) Malloc c) Free d) all of these
   Answer: (d)

4. Which type of linked List contains a pointer to the next as well as previous node in the sequence? [WBUT 2013]
   a) Singly Linked List b) Circular Linked List
   c) Doubly Linked List d) All of these
   Answer: (c)

5. Linked list is not suitable data structure for which one of the following problems? [WBUT 2014]
   a) insertion sort b) radix sort
   c) binary search d) polynomial addition
   Answer: (c)

6. Self-referential pointer is used in defining [WBUT 2015]
   a) an array b) a node of linked-list
   c) a queue d) all of these
   Answer: (d)

7. Inserting a node after a given node in a doubly linked list requires [WBUT 2016, 2018]
   a) four pointer exchanges b) two pointer exchanges
   c) one pointer exchange d) no pointer exchange
   Answer: (b)

8. Binary search is not possible for [WBUT 2017]
   a) array b) linked list c) stack d) queue
   Answer: (b)

9. A linear link list can be traversed using [WBUT 2017]
a) recursion b) both (a) and (c) are correct
c) stack d) both (a) and (c) are wrong

Answer: (a)

10. The data structure used to solve recursive problem is [WBUT 2017]
a) Linked list b) Queue c) Stack d) none of these

Answer: (c)

Short Answer Type Questions

1. Write an algorithm to add two polynomials. [WBUT 2007]

Answer:

Let us assume that the two polynomials are represented using linked list and the resultant is also using a linked list.

Let phead1, phead2 and phead3 represent the pointers of the three lists under consideration.

Let each node contain two integers: exp and coff.

Let us assume that the two linked lists already contain relevant data about the two polynomials.

Also assume that we have got a function append to insert a new node at the end of the given list.

```
p1 = phead1;
p2 = phead2;
```

Let us call malloc to create a new node p3 to build the third list

```
p3 = phead3;
/* now traverse the lists till one list gets exhausted */
while ((p1 != NULL) || (p2 != NULL))
{
 /* if the exponent of p1 is higher than that of p2 then the next term in final list is going to be the node of p1 */
   while (p1 ->exp   >  p2 -> exp )
     {
          p3 -> exp  = p1 -> exp;
          p3 -> coff = p1 -> coff ;
          append (p3, phead3);
         /* now move to the next term in list 1*/
         p1 = p1 -> next;
     }
 /* if p2 exponent turns out to be higher than make p3 same as p2 and append to final list */
while (p1 ->exp   <  p2 -> exp )
   {
        p3 -> exp  = p2 -> exp;
        p3 -> coff = p2 -> coff ;
```



```
        appe
        p2 =
    }
/* now consider th
coefficients to get t
while (p1 ->exp
   {
    p3-> exp =
    p3->coff =
          app
          p1
    p2 = p2->n
    }
}
/* now consider th
only in list1. So all
not have to do it te
append the pointer
 if ( p1 != NUL
      append (p
else
      append (p
```

2. Write the key fe
of Josephus probl
Answer:
A circular list node
a reference to its ta
head node. This mal
Josephus problem is
There are people s
executed, certain nu
people are skipped
(which is becoming
the last man remains
The task is to choos
remaining).
The easiest and mo
circular link list as s

```
/*linked list s
struct node {
      int item;
      node* nex
};
/* function*/
int simple_simu
```

```
        append (p3, phead3);
        p2 = p2 -> next;
    }
/* now consider the possibility that both exponents are same , then we must add the
coefficients to get the term for the final list */
while (p1 ->exp    =   p2 -> exp )
    {
    p3-> exp = p1-> exp;
    p3->coff = p1->coff + p2-> coff ;
           append (p3, phead3) ;
           p1 = p1->next ;
    p2 = p2->next ;
       }
}
/* now consider the possibility that list2 gets exhausted , and there are terms remaining
only in list1. So all those terms have to be appended to end of list3. However, one does
not have to do it term by term, as p1 is already pointing to remaining terms, so simply
append the pointer p1 to phead3 */
 if ( p1 != NULL)
     append (p1, phead3) ;
else
     append (p2, phead3);
```

2. Write the key features of circular linked list and state why it is important in case of Josephus problem. **[WBUT 2007]**

Answer:

A circular list node is identical to a singly-linked list node. However, the circular list has a reference to its tail node instead of its head node. The tail node has a reference to the head node. This makes it possible to get to both ends of the list in constant time.

Josephus problem is stated as:

There are people standing in a circle waiting to be executed. After the first man is executed, certain number of people are skipped and one man is executed. Then again, people are skipped and a man is executed. The elimination proceeds around the circle (which is becoming smaller and smaller as the executed people are removed), until only the last man remains, who is given freedom.

The task is to choose the place in the initial circle so that you survive (are the last one remaining).

The easiest and most logical way to solve the problem is to simply simulate it using a circular link list as shown below.

```
/*linked list structure */
struct node {
     int item;
     node* next;
};
/* function*/
int simple_simulation(int N, int M)
```

```
{
  int i;
  node *t=new node();
  node *x=t;
  t->item = 1; t->next = t;
  //Construct the list
  for (i = 2; i <= N; i++)
  {
    x = (x->next = new node ());
    x->item = i;
    x->next = t;
  }
  //Find the survivor
  while (x != x->next)
  {
    for (i = 1; i < M; i++)
      x = x->next;
    x->next = x->next->next; N--;
  }
  return x->item;
}
```

3. What are the advantages of linked list over an array? [WBUT 200
Write an algorithm to insert a data X after a specific data item Y in a linked list. [WBUT 2008, 2016, 20

Answer:

1st Part:

The advantages of linked list over an array are:

Refer to Question No. 5 of Short Answer Type Questions.

2nd Part:

Assuming the linked list already contains n integer elements. The algorithm to inser data X after a specific data item Y in a linked list is as follows:

```
node *insertXafterY(node *p, int y, int x)
{
  node *q,  *r,  *m;
  q = p;
  r = (node*)malloc(sizeof(node));
  while (p->next != NULL)
  {
    m = p;
    if (p->data == x)
    {
      break;
    }
    else
      p = p->next;
}
```

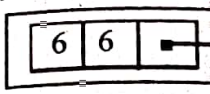


```
  r->data = y;
  r->next = m->next
  m->next = r;
  return (q);
}
```

4. How can a polynomi
list?

Answer:

We can represent the poly
a pointer to the next node



5. What are the advant
array?

Answer:

Linked list advantages

1. Linked lists do not
   stored in an array mi
2. Linked list storage
   contiguous memory
3. Inserting or removin
   or removing an elem
   need to be shifted).

Linked list disadvantag

1. Arrays have contigu
   between.
2. As memory is alloca
3. Fixed in size so if
   memory wastage wh
4. Insertion and deletio
5. Accessing data is ea

6. Write a C language
conditions are to be ha

Answer:

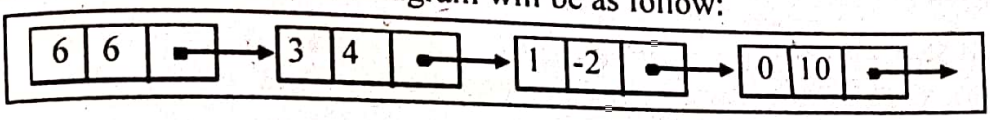
```
struct node* Delet
    struct node*
    int length =
    int i;
  if(n <= 0 || n
      printf("ER
  }else{
      if(n == 1)
```

```
r->data = y;
r->next = m->next;
m->next = r;
return (q);
}
```

4. How can a polynomial such as $6x^6 + 4x^3 - 2x + 10$ be represented by a linked list? **[WBUT 2010, 2015]**

Answer:

We can represent the polynomial by a linked list, where each node contains deg, coef and a pointer to the next node. So, the diagram will be as follow:



5. What are the advantages and disadvantages of linked list data structure over array? **[WBUT 2010]**

Answer:

Linked list advantages over array

1. Linked lists do not need contiguous blocks of memory; extremely large data sets stored in an array might not be able to fit in memory.
2. Linked list storage does not need to be preallocated (again, due to arrays needing contiguous memory blocks).
3. Inserting or removing an element into a linked list requires one data update, inserting or removing an element into an array requires n (all elements after the modified index need to be shifted).

Linked list disadvantages over array:

1. Arrays have contiguous memory allocation which makes it easy to access elements in between.
2. As memory is allocated during compilation makes the program faster
3. Fixed in size so if we are aware of the exact size of datas then there can be no memory wastage which is also an advantage linked list has.
4. Insertion and deletion at the end of the array is easy but not in between.
5. Accessing data is easy. Example, a[2].

6. Write a C language function to delete nth node of a singly-linked list. The error conditions are to be handled properly. **[WBUT 2011]**

Answer:

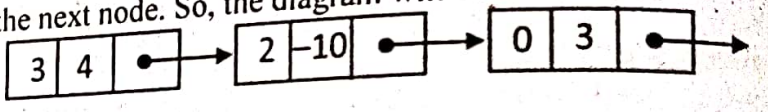
```
struct node* DeleteNode(struct node* head, int n) {
    struct node* temp = head;
    int length = LinkedListLength(temp);
    int i;
   if(n <= 0 || n > length){
      printf("ERROR: Node does not exist!\n");
   }else{
      if(n == 1){
```

```
            head = head->next; //move from head (1st node) to
second node
        }else{
            for(i = 1; i < n-1; ++i){ //move through list
                temp = temp->next;

            }
            temp->next = temp->next->next;

        }
    }
    return head;
}
```

7. a) How the polynomial $4x^3 - 10x^2 + 3$ can be represented using a linked list?
b) Compare and contrast between an array and a single linked list. [WBUT 20

Answer:

a) We can represent the polynomial by a linked list, where each node contains deg, c and a pointer to the next node. So, the diagram will be as follow:



b) *Refer to Question No. 5 of Short Answer Type Questions.*

8. What is a self referential structure? [WBUT 20

Answer:

A self-referential structure is one of the data structures which refer to the pointe (points) to another structure of the same type. For example, a linked list is supposed to a self-referential data structure. The next node of a node is being pointed, which is of same struct type. For example,

```
typedef struct listnode {
void *data;
struct listnode *next;
} linked_list;
```

In the above example, the listnode is a self-referential structure – because the *next s the type sturct listnode.

9. Write an algorithm to find the largest and smallest element in a single linear li [WBUT 20

Answer:

Linked List definition in C:

```
typedef struct linked_list
{
    int data;
    struct linked_list *next;
} Node;
```

Code to find largest and smallest elements.

```
// finds the largest and smallest in the list
// assumes that head pointer is defined elsewhere
```



```
int MaxMinInLis
{
    // start at
    Node *curre
    if (current
        return
    // initiali
    *max = *min
    // loop thr
    for (curren
currentNode = c
    {
        if (cur
            *ma
        else if
            *mi
    }
    // we found
    return 1;
}
```

10. Write an algori

Answer:

C function to delete

```
struct node *de
{
  struct node
  struct node
  if(head->next
  {
    free(head)
    head=NULL;
  }
  else
  {
     while(temp
     {
        t=temp
        temp=t
     }
     free(t->n
     t->next=NU
  }
  return head;
}
```

11. Write an algori

```
int MaxMinInList(int *max, int *min)
{
   // start at the root
   Node *currentNode = head;
   if (currentNode == NULL)
       return 0; // list is empty
   // initialize the max and min values to the first node
   *max = *min = currentNode->data;
   // loop through the list
   for (currentNode = currentNode->next; currentNode != NULL;
currentNode = currentNode->next)
   {
       if (currentNode->data > *max)
           *max = currentNode->data;
       else if (currentNode->data < *min)
           *min = currentNode->data;
   }
   // we found our answer
   return 1;
}
```

10. Write an algorithm to delete the last node of a linked-list. **[WBUT 2015]**

Answer:

C function to delete last node is as given below:

```
struct node *delete(struct node *head)
{
  struct node *temp =head;
  struct node *t;
  if(head->next==NULL)
  {
    free(head);
    head=NULL;
  }
  else
  {
     while(temp->next != NULL)
     {
        t=temp;
        temp=temp->next;
     }
     free(t->next);
     t->next=NULL;
  }
  return head;
}
```

11. Write an algorithm to insert an element in the middle of a linked list. **[WBUT 2015]**

Answer:

Node Structure for Singly Linked List:

```
struct node {
        int data;
        struct node *next;
}start = NULL;
```

Insert node at middle position : Singly Linked List

```
void insert_mid()
{
    int pos,i;
    struct node *new_node,*current,*temp,*temp1;
    new_node=(struct node *)malloc(sizeof(struct node));
    printf("nEnter the data : ");
    scanf("%d",&new_node->data);
    new_node->next=NULL;
    st :
    printf("nEnter the position : ");
    scanf("%d",&pos);
    if(pos>=(length()+1))
       {
       printf("nError : pos > length ");
       goto st;
       }
    if(start==NULL)
       {
       start=new_node;
       current=new_node;
       }
    else
       {
       temp = start;
             for(i=1;i< pos-1;i++)
             {
             temp = temp->next;
             }
       temp1=temp->next;
       temp->next = new_node;
       new_node->next=temp1;
       }
}
```

12. What is the difference between array and linked-list? [WBUT 2

Answer:

The primary difference between an array or ArrayList and a linked list is that the arr
ArrayList permits random access into the list structure using a subscript; each data
has a "named" storage location (named by the subscript). In contrast, with a linked li
locations are relative to the next or previous locations. With an array or ArrayList, w
access the 10^{th} data item, say, with one probe to subscript 9 (indexing zero-up).

linked list, we cannot ac
Searching through a linke
On the other hand, becau
list can be made withou
insertions and deletions
exactly as needed; no da
left. The deleted node bec

13. Show how linked lis

$5x^4 + 5x^3 + 10x^2 + 8$
$3x^3 + 2x^2 + 7x + 8$.

Answer:

Let us assume that the tw
the resultant is also using
Let each node contain tw
The two linked lists alr
create a linked list list3 (

Steps: We traverse the l

1) if the exponent
going to be the
2) if p2 exponent
final list
3) now consider th
coefficients to g
4) now consider th
remaining only

The final solution wou
algorithm.

14. How a polynomial
linked list? What are
array?

Answer:

1st part:

We can represent the p
a pointer to the next no
5|8 →3|5→ 2|-9→1|2→

2nd part: ***Refer to Qu***

linked list, we cannot access the 10^{th} item without first traversing the first nine items. Searching through a linked list is thus inherently a sequential process.
On the other hand, because all locations are relative, additions and deletions to a linked list can be made without affecting any of the data items already stored in the list. insertions and deletions require only that the next and previous pointers be changed exactly as needed; no data needs to be moved. When a deletion takes place, no hole is left. The deleted node becomes unused memory.

13. Show how linked list can be used to add the following polynomials:
$5x^4 + 5x^3 + 10x^2 + 8x + 3$
$3x^3 + 2x^2 + 7x + 8$.
[WBUT 2016]

Answer:
Let us assume that the two polynomials (p1 and p2) are represented using linked list and the resultant is also using a linked list.
Let each node contain two integers: exp and coff.
The two linked lists already contain relevant data about the two polynomials. We will create a linked list list3 (p3) with a single node to begin with.
Steps: We traverse the lists till one list gets exhausted

1) if the exponent of p1 is higher than that of p2 then the next term in final list is going to be the node of p1
2) if p2 exponent turns out to be higher than make p3 same as p2 and append to final list
3) now consider the possibility that both exponents are same , then we must add the coefficients to get the term for the final list
4) now consider the possibility that list2 gets exhausted , and there are terms remaining only in list1. So all those terms have to be appended to end of list3.

The final solution would be 5x^4 + 8x^3 + 12x^2 + 15x + 11 based on the above algorithm.

14. How a polynomial such as $8x^5 + 4x^3 - 9x^2 + 2x - 17$ can be represented using a linked list? What are the advantages and disadvantages of linked list over an array? **[WBUT 2017]**

Answer:
1st part:
We can represent the polynomial by a linked list, where each node contains deg, coef and a pointer to the next node. So, the diagram will be as follow:
5|8 →3|5→ 2|-9→1|2→0|-17

2nd part: ***Refer to Question No. 5 of Short Answer Type Questions.***

15. Compare and contrast linked list with static and dynamic array. [WBUT 2010]

Answer:

In contrast, linked lists are dynamic and flexible and it can expand and contract is size. In an static array, memory is assigned during compile time. While in a dynamic array and linked list, it is allocated during execution or runtime. Elements are stored consecutively in arrays whereas it is stored randomly in linked lists.

Compare:

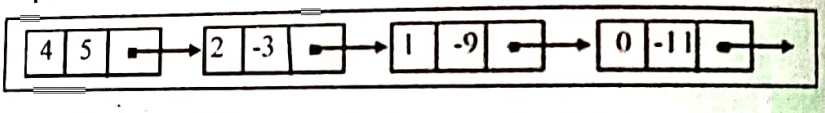
1. An array is the data structure that contains a collection of similar type data elements whereas the linked list is considered as non–primitive data structure contains a collection of unordered linked elements known as nodes.
2. Accessing an element in an array is fast, while linked list takes linear time, so it is quite a bit slower.
3. Arrays are of fixed size. Linked lists are dynamic and flexible and can expand and contract its size.
4. In addition memory utilization is inefficient in an array. Conversely, memory utilization is efficient in the linked list.

Long Answer Type Questions

1. a) How can a polynomial such as $5x^4 - 3x^2 + 9x - 11$ be represented by a linked list? [WBUT 2007]

Answer:

We can represent the polynomial by a linked list, where each node contains deg, coefficient and a pointer to the next node. So, the diagram will be as follow:



b) Write an algorithm to delete a node from a doubly linked list, where a node contains one data and two address (prev & next) portion. [WBUT 2007]

OR,

Write an algorithm for deletion of a node from a doubly-linked list. [WBUT 2011]

Answer:

```
/*C function*/
void deleteNode(node *n)
{
      node *np = n->prev;
      node *nn = n->next;
      np->next = n->next;
      nn->prev = n->prev;
      delete n;
}
```

2. a) Do you consider the following list.

b) Represent the following po

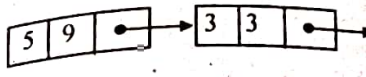
$9x^5 + 3x^3 - 8x + 15$.

c) Write an algorithm to dele singly linked list.

Answer:

a) Linear, because traverse all

b)



c)

```
node *removeNodeAfterX(
{
  // r holds the addres
  node *q,  *r,  *m;
  q = p;
  while (q!= NULL)
  {
      m = q;
      if(q->next == NU
      {
          if (q->data
          {
              r->next =
              free(q);
          }
          break;
      }
      else if (q->data
      {
        r->next = m->n
        free(q);
        q = r;
      }
      else
      {
          r = q;
          q = q->next;
      }
  }
}
  return q;
```

2. a) Do you consider the following data-structure as linear? Circular doubly linked list.

b) Represent the following polynomial by linked list (show the diagram only)

$9x^5 + 3x^3 - 8x + 15$.

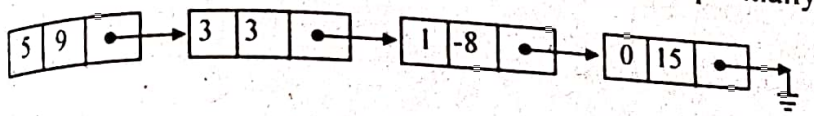
c) Write an algorithm to delete all nodes having value greater than X from a given singly linked list.

[WBUT 2008]

Answer:

a) Linear, because traverse all the list of elements sequentially.

b)



c)

```
node *removeNodeAfterX(node *p, int x)
{
 // r holds the address of the previous node
 node *q,   *r,   *m;
 q = p;
 while (q!= NULL)
 {
    m = q;
    if(q->next == NULL)
    {
        if (q->data > x)
        {
            r->next = NULL;
            free(q);
        }
        break;
    }
    else if (q->data > x)
    {
      r->next = m=>next;
      free(q);
      q = r;
    }
    else
    {
       r = q;
       q = q->next;
    }
 }
 return q;
}
```

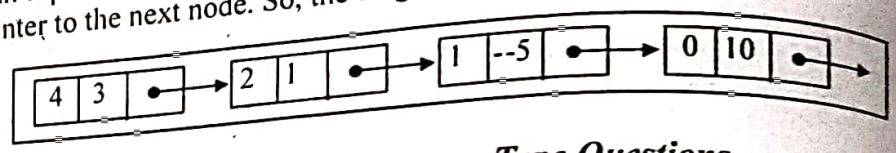
3. a) Represent the given polynomial using a link list. [WBUT 20

$3x^4 + x^2 - 5x + 2$

b) Write the pseudo code / C code for adding two polynomials (already given user, no need to take input). Also comment on the complexity of your algorithm.

Answer:

a) We can represent the polynomial by a linked list, where each node contains deg and a pointer to the next node. So, the diagram will be as follow:



b) *Refer to Question No. 1 of Short Answer Type Questions.*

Running is $\Theta(\max(x,y)\cdot(x+y))$ where x and y are the number of terms in the polynom Poly1 and Poly2 respectively.

4. a) Write an algorithm for creating a linked-list with n nodes. [WBUT 20

b) How it can be made a circular linked-list? Write a function for that purpose.

Answer:

a) //C functions showing how to create a linked list with n nodes with values 1 to n sequence

```
#include<stdio.h>
#include<stdlib.h>

// A linked list node
struct Node
{
  int data;
  struct Node *next;
};

/* Given a reference (pointer to pointer) to the head
   of a list and an int, appends a new node at the end  */
void append(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(str
Node));
    struct Node *last = *head_ref;  /* used in step 5*/

    /* 2. put in the data  */
    new_node->data  = new_data;

    /* 3. This new node is going to be the last node, so
next of
          it as NULL*/
    new_node->next = NULL;
```

```
    /* 4. If the
head */
    if (*head_ref
    {
        *head_ref =
        return;
    }

    /* 5. Else tra
    while (last->n
        last = las

    /* 6. Change t
    last->next = n
    return;
}
/*this creates a l
int createList()
{
  /* Start with th
  struct Node* hea
for (i=0; i<n; i++
// Insert data at
  append(&head, i);
}
```

b)

```
// Function tha
// into circular l
struct Node* circu
{
    // declare a n
    // assign head
    struct Node* s

    // check that
    // to NULL the
    while (head->n
        head = hea

    // if head->ne
    // start assig
    head->next = s
    return start;
}
```

```
    /* 4. If the Linked List is empty, then make the new node as
head */
    if (*head_ref == NULL)
    {
        *head_ref = new_node;
        return;
    }

    /* 5. Else traverse till the last node */
    while (last->next != NULL)
        last = last->next;

    /* 6. Change the next of last node */
    last->next = new_node;
    return;
}
/*this creates a list*/
int createList()
{
  /* Start with the empty list */
  struct Node* head = NULL;
for (i=0; i<n; i++)
// Insert data at the end
  append(&head, i);
}
```

b)

```
// Function that convert singly linked list
// into circular linked list.
struct Node* circular(struct Node* head)
{
    // declare a node variable start and
    // assign head node into start node.
    struct Node* start = head;

    // check that while head->next not equal
    // to NULL then head points to next node.
    while (head->next != NULL)
        head = head->next;

    // if head->next points to NULL then
    // start assign to the head->next node.
    head->next = start;
    return start;
}
```

5. How a linked-lists can be used to implement stack? [WBUT 20

Answer:

To implement stack using linked list, we need to set the following things befo implementing actual operations.

Step 1: Define a 'Node' structure with two members data and next.

Step 2: Define a Node pointer 'top' and set it to NULL.

Step 3: Implement the main method by displaying Menu with list of operations and ma suitable function calls in the main method.

push(value) - Inserting an element into the Stack

We can use the following steps to insert a new node into the stack...

Step 1: Create a newNode with given value.

Step 2: Check whether stack is Empty (top == NULL)

Step 3: If it is Empty, then set newNode → next = NULL.

Step 4: If it is Not Empty, then set newNode → next = top.

Step 5: Finally, set top = newNode.

pop() - Deleting an Element from a Stack

We can use the following steps to delete a node from the stack...

Step 1: Check whether stack is Empty (top == NULL).

Step 2: If it is Empty, then display "Stack is Empty!!! Deletion is not possible!!!" a terminate the function

Step 3: If it is Not Empty, then define a Node pointer 'temp' and set it to 'top'.

Step 4: Then set 'top = top → next'.

Step 7: Finally, delete 'temp' (free(temp)).

6. Write short notes on the following:

a) Linear Lists [WBUT 20

b) Circular link list [WBUT 20

Answer:

a) Linear Lists:

A **linear list** stores a collection of objects of a certain type, usually denoted as **elements** of the list. The elements are ordered within the linear list in a linear sequen Linear lists are usually simply denoted as **lists.**

Unlike an array, a list is a data structure allowing insertion and deletion of elements a arbitrary position of the sequence. If the position in question is given, for example b reference, such a modification takes *only a constant number* of operations, that is effortful copying of entries is necessary and all insertion and deletion operations take equally short time. Conversely, however, one cannot access a single element via (integral) index in constant time, as in the case of an array, without having searched fo before and having received a reference to it. Furthermore, lists are not limited to a cer maximum number of elements from the beginning on (like an array).

b) Circular link list:

Circular Linked List is

- Singly
- Doubly

- In Circular Li Node".
- In short First N
- Linked List is chain as shown

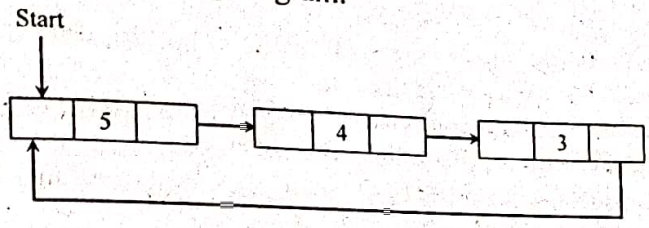
- Two way acce
- Sequential mo
- No direct acce

b) Circular link list:

Circular Linked List is Divided into 2 Categories.

- Singly Circular Linked List
- Doubly Circular Linked List

- In Circular Linked List Address field of Last node contain address of "First Node".
- In short First Node and Last Nodes are adjacent.
- Linked List is made circular by linking first and last node, so it looks likecircular chain as shown in following diagram.



- Two way access is possible only if we are using "Doubly Circular Linked List"
- Sequential movement is possible
- No direct access is allowed.