1) C Program to Concatenate two singly linked lists ?

Ans :

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int info;
	struct node *link;
};
struct node *create_list(struct node *);
struct node *concat( struct node *start1,struct node *start2);
struct node *addatbeg(struct node *start, int data);
struct node *addatend(struct node *start,int data);
void display(struct node *start);
int main()
{
	struct node *start1=NULL,*start2=NULL;
	start1=create_list(start1);
	start2=create_list(start2);
	printf("\nFirst list is  : ");
	display(start1);
	printf("\nSecond list is  : ");
	display(start2);
    start1=concat(start1, start2);
	printf("\nConcatenated list is  : ");
	display(start1);
	return 0;
}
struct node *concat( struct node *start1,struct node *start2)
{
	struct node *ptr;
	if(start1==NULL)
	{
		start1=start2;
		return start1;
	}
	if(start2==NULL)
		return start1;
	ptr=start1;
	while(ptr->link!=NULL)
		ptr=ptr->link;
```

```
	ptr->link=start2;
	return start1;
}
struct node *create_list(struct node *start)
{
	int i,n,data;
	printf("\nEnter the number of nodes : ");
	scanf("%d",&n);
	start=NULL;
	if(n==0)
		return start;

	printf("Enter the element to be inserted : ");
	scanf("%d",&data);
	start=addatbeg(start,data);

	for(i=2;i<=n;i++)
	{
		printf("Enter the element to be inserted : ");
		scanf("%d",&data);
		start=addatend(start,data);
	}
	return start;
}
void display(struct node *start)
{
	struct node *p;
	if(start==NULL)
	{
		printf("\nList is empty\n");
		return;
	}
	p=start;
	while(p!=NULL)
	{
		printf("%d ", p->info);
		p=p->link;
	}
	printf("\n");
}
struct node *addatbeg(struct node *start,int data)
{
	struct node *tmp;
	tmp=(struct node *)malloc(sizeof(struct node));
```

```
        tmp->info=data;
        tmp->link=start;
        start=tmp;
        return start;
}
struct node *addatend(struct node *start, int data)
{
        struct node *p,*tmp;
        tmp= (struct node *)malloc(sizeof(struct node));
        tmp->info=data;
        p=start;
        while(p->link!=NULL)
                p=p->link;
        p->link=tmp;
        tmp->link=NULL;
        return start;
}
```

```
<terminated> (exit value: 0) mergelinkedlist [C/C++ Application]

Enter the number of nodes : 5
Enter the element to be inserted : 3
Enter the element to be inserted : 4
Enter the element to be inserted : 1
Enter the element to be inserted : 4
Enter the element to be inserted : 6

Enter the number of nodes : 3
Enter the element to be inserted : 6
Enter the element to be inserted : 9
Enter the element to be inserted : 8

First list is  : 3 4 1 4 6

Second list is  : 6 9 8

Concatenated list is  : 3 4 1 4 6 6 9 8
```

MERGING OF TWO LINKED LIST

Algorithm

(1) set ptr = head1
(2) while(ptr->next != NULL)
{
ptr = ptr->next;
}
(3) ptr->next = head2
(4) Return(head2)

2) Write a C program to to Copy one Linked List to Another ?

Ans :

```
#include <stdio.h>
#include <stdlib.h>
struct Node {

    int data;
```

```
    struct Node* next;
};
void printList(struct Node* head)
{
    struct Node* ptr = head;
    while (ptr) {
        printf("%d -> ", ptr->data);
        ptr = ptr->next;
    }
    printf("NULL");
}
void insert(struct Node** head_ref, int data)
{
    struct Node* newNode
        = (struct Node*)malloc(
            sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head_ref;
    *head_ref = newNode;
}
struct Node* copyList(struct Node* head)
{
    if (head == NULL) {
        return NULL;
    }

    else {
        struct Node* newNode
            = (struct Node*)malloc(
                sizeof(struct Node));
        newNode->data = head->data;
        newNode->next = copyList(head->next);
        return newNode;
    }
}
struct Node* create(int arr[], int N)
{
    struct Node* head_ref = NULL;
    for (int i = N - 1; i >= 0; i--) {
        insert(&head_ref, arr[i]);
    }
    return head_ref;
}
void printLists(struct Node* head_ref,
```

```
                struct Node* dup)
{
    printf("Original list: ");
    printList(head_ref);
    printf("\nDuplicate list: ");
    printList(dup);
}
int main(void)
{
    int arr[] = { 1, 2, 3, 4, 5 };
    int N = sizeof(arr) / sizeof(arr[0]);
    struct Node* head_ref = create(arr, N);
    struct Node* dup = copyList(head_ref);
    printLists(head_ref, dup);
    return 0;
}
```

```
<terminated> (exit value: 0) copylist [C/C++ Application] /home/akhil/
Original list: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Duplicate list: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
```

COPY OF LINKED LIST

1) IF(Ptr == NULL)

2) List is Empty

3) Else

4) ptr = head->next

5) ptr1 = head1->next

6) while(ptr != NULL)

{

ptr->data = ptr->data

ptr = ptr->next

ptr1 = ptr1 = next

}

7 ptr1->next = NULL

3)Write a program to Find the no.of nodes in a linked list ?

Ans :

```
    #include<stdio.h>
#include<stdlib.h>
struct Node
{
   int data;
   struct Node* next;
};
void push(struct Node** head_ref, int new_data)
{
   struct Node* new_node =
          (struct Node*) malloc(sizeof(struct Node));
   new_node->data  = new_data;
   new_node->next = (*head_ref);
   (*head_ref)    = new_node;
}
int getCount(struct Node* head)
{
   int count = 0;
   struct Node* current = head;
   while (current != NULL)
   {
      count++;
      current = current->next;
   }
   return count;
}
int main()
{
   struct Node* head = NULL;
   /* Use push() to construct below list
    1->2->1->3->1  */
   push(&head, 1);
   push(&head, 3);
   push(&head, 1);
   push(&head, 2);
   push(&head, 1);
   printf("count of nodes is %d", getCount(head));
   return 0;
}
```

```
<terminated> (exit value: 0) noofnode [C/
count of nodes is 5
```

4) Algorithms of tree traversal- inorder, post order , post order ?

Ans :

PREORDER TRAVERSAL

1) ptr = Root
2) if(ptr ≠ null) then
3) visit(ptr)
4) preorder(ptr → LC)
5) preorder(ptr → RC)
6) ENDIF

INORDER TRAVERSAL

1) ptr = Root
2) IF (ptr ≠ Null) then
3) inorder (ptr → LC)
4) visit (ptr)
5) inorder (ptr → RC)
6) ENDIF

POST ORDER TRAVERSAL

6) ENDIF

POST ORDER TRAVERSAL

1) ptr = Root
2) if (ptr != NULL) then
3) POSTORDER(ptr -> LC)
4) POSTORDER (ptr -> RC)
5) visit (ptr)
6) ENDIF

5) Write a program to Splitting a linked list ?

```
#include <stdio.h>
#include <stdlib.h>
struct node {
   int data;
   struct node *next;
};
struct node *even = NULL;
struct node *odd = NULL;
struct node *list = NULL;
void insert(int data) {
   struct node *link = (struct node*) malloc(sizeof(struct node));
   struct node *current;
   link->data = data;
   link->next = NULL;
   if(list == NULL) {
      list = link;
      return;
```

```
  }
  current = list;
  while(current->next!=NULL)
    current = current->next;
  current->next = link;
}
void display(struct node *head) {
  struct node *ptr = head;
  printf("[head] =>");
  while(ptr != NULL) {
    printf(" %d =>",ptr->data);
    ptr = ptr->next;
  }
  printf(" [null]\n");
}
void split_list() {
  struct node *link;
  struct node *current;
  while(list != NULL) {
    struct node *link = (struct node*) malloc(sizeof(struct node));
    link->data = list->data;
    link->next = NULL;
    if(list->data%2 == 0) {
      if(even == NULL) {
        even = link;
        list = list->next;
        continue;
      } else {
        current = even;
        while(current->next != NULL)
        current = current->next;
        current->next = link;
      }
      list = list->next;
    } else {
      if(odd == NULL) {
        odd = link;
        list = list->next;
        continue;
      } else {
        current = odd;
        while(current->next!=NULL)
        current = current->next;
        current->next = link;
```

```
        }
        list = list->next;
      }
    }
}
int main() {
   int i;
   for(i = 1; i <= 10; i++)
      insert(i);
   printf("Complete list: \n");
   display(list);
   split_list();
   printf("\nOdd  : ");
   display(odd);
   printf("Even : ");
   display(even);
   return 0;
}
```

```
Complete list:
[head] => 1 => 2 => 3 => 4 => 5 => 6 => 7 => 8 => 9 => 10 => [null]

Odd  : [head] => 1 => 3 => 5 => 7 => 9 => [null]
Even : [head] => 2 => 4 => 6 => 8 => 10 => [null]
```

6) Implement a program Circular queue using circular linked list ?

Ans :

```
#include<stdio.h>
#include<stdlib.h>
#define que struct queue
#define pf printf
#define sf scanf
struct queue{
int info;
struct queue *link;
};
 que *front=NULL,*rear=NULL;
int count=0;
void push(int n)
{
```

```
que *newnode;
newnode=(struct queue*)malloc(sizeof(struct queue));
newnode->info=n;
newnode->link=NULL;
if(count==0)
front=newnode;
else
        rear->link=newnode;
    rear=newnode;
    rear->link=front;
count++;
}
int pop(void)
{
int n;
que *temp;
if(count==0)
return (-1);
count--;
    if(front==rear)
    {
        n=front->info;
        free(front);
        front=NULL;
        rear=NULL;
    }else
    {
            temp= front ;
            n = temp-> info ;
            front = front -> link ;
            rear -> link = front ;
            free ( temp ) ;
    }
return n;
}
void display(void)
{
que *temp;
int i;
if(count==0)
pf("Empty");
else
{
temp=front;
```

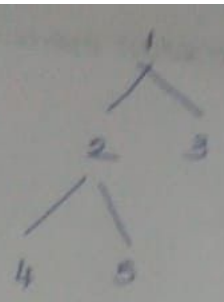
```
for(i=0;i<count;i++)
{
pf("%d ",temp->info);
temp=temp->link;
}
}
pf("\n");
}
int size(void)
{
return count;
}
int main()
{
int n,ch=10;
while(ch!=0)
{
pf("\n       What do you want to do??\n");
pf("1.Push\n");
pf("2.Pop\n");
pf("3.SizeOfQueue\n");
pf("4.Display\n");
pf("0.EXIT\n");
sf("%d",&ch);
switch(ch)
{
case 1:
{
pf("What no. do you want to push in queue\n");
sf("%d",&n);
push(n);
break;
}
case 2:
{
n=pop();
if(n==-1)
pf("Queue is empty\n");
else
pf("Number poped from queue is %d\n",n);
break;
}
case 3:
{
```

```
n=size();
pf("Size of queue is %d\n",n);
break;
}
case 4:
{
pf("Queue is -->> ");
display();
}
case 0:
break;
default:
pf("Wrong Choice\n");
break;
}
}
}
```

```
w [C/C++ Application]

       What do you want to do??
1.Push
2.Pop
3.SizeOfQueue
4.Display
0.EXIT
1
What no. do you want to push in queue
10

       What do you want to do??
1.Push
2.Pop
3.SizeOfQueue
4.Display
0.EXIT
1
What no. do you want to push in queue
20

       What do you want to do??
1.Push
2.Pop
3.SizeOfQueue
4.Display
0.EXIT
2
Number poped from queue is 10

       What do you want to do??
```

```
       What do you want to do??
1.Push
2.Pop
3.SizeOfQueue
4.Display
0.EXIT
4
Queue is -->> 20

       What do you want to do??
1.Push
2.Pop
3.SizeOfQueue
4.Display
0.EXIT
0
```

7) Inorder Traversal algorithm without recursion by considering an Example



Step 1: creates an empty stack: S=NULL

Step 2: set current as adress of root: current => 1

Step 3: pushes the current node and set current =
current -> left until current is NULL

current -> 1
push 1: stack S -> 1
current: -> 2
push 2: stack S -> 2, 1
current -> 4
push 4: stack S -> 4, 2, 1
current = NULL

step 4 pops from S

a) pop 4: stack S -> 2, 1.
b) print "4"
c) current = NULL /* right of 4 */ and go to step 3
since current is NULL step 3 doesn't do anything

step 4 pops again.

a) pops: stack S -> 1
b) print "2"
c) current -> 5 /* right of 2 */ and go to step 3

Step 3: pushes 5 to stack and makes current NULL

Stack S -> 5, 1

Current = NULL

Step 4 pops from S

a pops 5: stack S -> 1

b) print "5"

c) current = NULL and go to step 3

Since current is NULL step 3 doesn't do anything

Step 4 pops Again

a) pop 1: stacks -> NULL

b) print "1";

c) current -> 3 /*right of 1*/

Step 3 pushes 3 to stack and makes current NULL

stacks -> 3

current = NULL

Step 4 pops from S

a) pop 3: stack S -> NULL

b) print "3"

c) current = NULL /*right of 3*/

Traversal is done now & stack S is empty and current is null

8)

Q

1) Height of the Tree T : = 3

2) Level(H) = 2
   Level(C) = 1
   Level(K) = 3

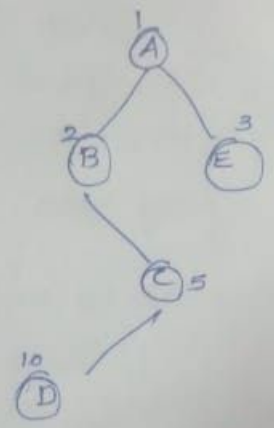
3) Degree of T = 12

4) The logest path
   A → C → F → g

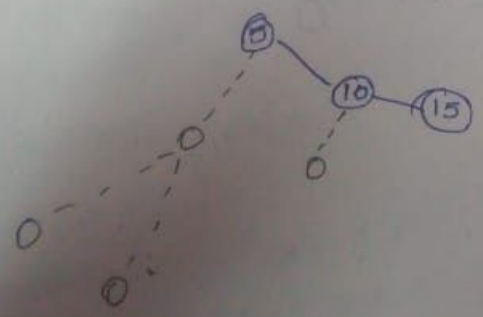
5 parent (m) => H
sibling (I) => J
child (B) => E, F, G

* Draw the tree structure whose array representation is given

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	E							D		C			



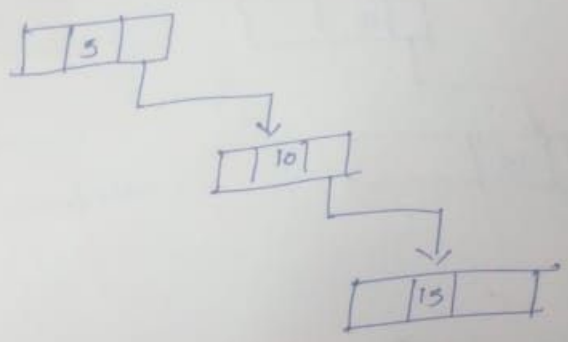
Q) Represent using array and linked list



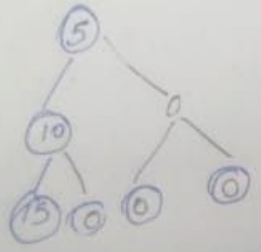
Array representation

1	2	3	4	5	6	7	8	9	10	11	12	13	14	25
5		10				15								

linked list Representation



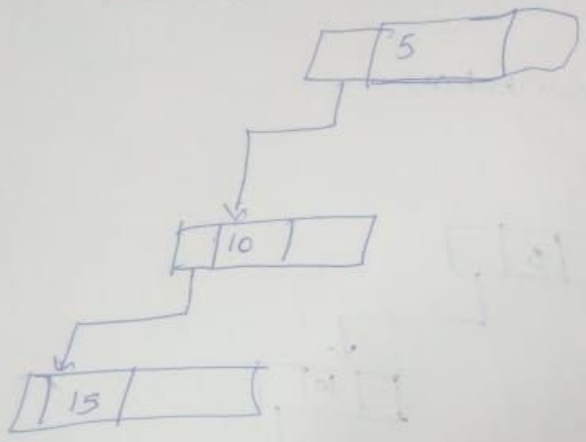
Q)



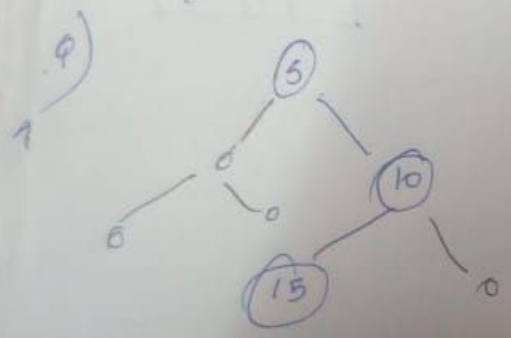
array Representation

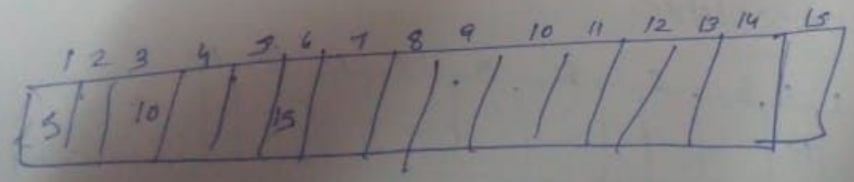
1	2	3	4	5	6	7	8	9	11	12	13	14
5	10		15									

Linked list



a)





Linked list