Shubham Garg
9919103057

DOI 15 | Page No.
Date / /

Tutorial-7

Program-1

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

struct Node
{   int data;
    struct Node *next;
};

int length(struct Node *p)
{   int l;
    while(p!=NULL)
    {   l++; p=p->next;
    }
    return l;
}

int Intersection(struct Node *p, struct Node *q)
{   int a,b;
    a=len(p);
    b=len(q);
    int flag=0;
    if(a>b
    {   int t=a-b;
        for(flag=0; flag<t; flag++)
            p=p->next;
    }
    else
    {   int t=b-a;
```

```
    for(flag=0; flag<t; flag++)
        q=q->next;
}
while(p!=NULL && q!=NULL)
{
    if(p->next==q->next)
        return p->data;
    p=p->next;
    q=q->next;
}
return -1;
}

void main()
{
    struct Node *t;
    struct Node *head1=(struct Node *)malloc(sizeof(struct Node
    head1->data=9;
    struct Node *head2=(struct Node *)malloc(sizeof(struct Node));
    head2->data=4;
    t=(struct Node *)malloc(sizeof(struct Node));
    t->data=6;
    head2->next=t;
    t=(struct Node *)malloc(sizeof(struct Node));
    t->data=2;
    head2->next->next=t;
    t=(struct Node *)malloc(sizeof(struct Node));
    t->data=15;
    head1->next=t;
    head2->next->next->next=t;
    head1->next->next=NULL;
    printf("Node Intersection is %d", Intersection(head1,head2));
}
```

Program 2

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{   int data;
    struct Node *next;
};
void find ( int x, int y, struct Node *he)
{   struct Node * curr, *temp, * c1, *c2, *p1, *p2;
    struct Node * prev = NULL;
    curr = he;
    while ( curr != NULL)
    {   if ( curr -> data = x)
        {   c1 = curr;
            p1 = prev;
        }
        if ( curr -> data = y)
        {   c2 = curr;
            p2 = prev;
        }
        prev = curr;
        curr = curr -> next;
    }
    printf ("%d %d %d %d", c1 -> data, c2 - data,
                           p1 -> data, p2 -> data);
    temp = c1 -> next;
    c1 -> next = c2 -> next;
    c2 - next = temp;
    p2 -> next = c1;
    p1 -> next = c2;
};
```

```
void main()
{
  void find(
  int i, size, item, n, item2, pi;
  struct Node * head;
  head = NULL;
  printf("Enter number of elements");
  scanf("%d", &n);
  struct Node * temp, * current;
  size = sizeof(struct Node);
  for(i=0; i<n; i++)
  {
     temp = (struct Node *) malloc(size);
     printf("Enter element\n");
     scanf("%d", &item);
     temp -> data = item;
     temp -> next = NULL;
     if(head == NULL)
     { head = temp; }
     else
        current -> next = temp;
     current = temp;
  }
  current = head;
  while(current != NULL)
  { printf("%d\n", current -> data);
     current = current -> next;
  }
  printf("Enter the keys to swap");
  int a, b;
  scanf("%d %d\n", &a, &b);
  find(a, b, head);
```

```
..current = head;
while (current != NULL)
{
    printf ("%d \n", current->data);
    current = current->next;
}
} // end of main
```

program 3

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

struct Node
{
   int data;
   struct Node *next;
} *first = NULL;

void create( int A[], int n)
{
   int i;
   struct Node *t, *last;
   first = (struct Node *)malloc(sizeof(struct Node));
   first->data = A[0];
   first->next = NULL;
   last = first;
   for(i=1; i<n; i++)
   {
      t = (struct Node *)malloc(sizeof(struct Node));
      t->data = A[i];
```

```
        t->next = NULL;
        last->next = t;
        last = t;
    }
}

void Reverse(struct Node *p)
{
    struct Node *q = NULL, *r = NULL;
    while(p != NULL)
    {
        r = q;
        q = p;
        p = p->next;
        q->next = r;
    }
    first = q;
}
int main()
{
    int A[] = {10, 20, 30, 40, 50};
    Create(A, 5);
    Reverse(first);
    return 0;
}
```

DOMS Page No.
Date / /

Program 4

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

struct Node
{   int data;
    struct Node * next;
};

struct node * newnode (int d);
{
   struct Node * newnode = newNode();
   newnode -> data = d;
   newnode -> next = NULL;
   return newnode;
}

struct node * reverse(struct Node * head, struct Node *prev)
{
   if(head == NULL)
       return NULL;
   struct Node * temp, * curr;
   temp= curr = head;
   while( curr != NULL && curr -> data % 2 == 0)
   {   temp = curr -> next;
       curr -> next = prev;
       prev = curr;
       curr = temp;
   }
```

```
if(curr != Head)
{ head->next = curr;
  curr = reverse(curr, NULL);
  return prev;
}
else
{ head->next = reverse(head->next, head);
  return head;
}
}
void main()
{
int a[] = {1,2,3,3,4,6,8,5};
int n = sizeof(a) / sizeof(a[0]);
struct node * head = NULL;
struct node * p;
for(int i=0; i<n; i++)
{
  if(head == NULL)
  { p = newnode(a[i]);
    head = p;
  }
  p->next = newnode(a[i]);
  p = p->next;
}
head = reverse(head, NULL);
while(head != NULL)
{ printf("%d \n", head->data);
  head = head->next;
}
return 0;
}
```

DOM'S Page No.
Date / /

Program 5

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{   int data;
    struct Node *next;
} *first=NULL;
void create(int A[], int n)
{   int i;
    struct Node *t, *last;
    first=(struct Node *)malloc(sizeof(struct Node));
    first->data=A[0];
    first->next=NULL;
    last=first;
    for(i=1;i<n;i++)
    {
      t=(struct Node*) malloc(sizeof(struct Node));
      t->data=A[i];
      t->next=NULL;
      last->next=t;
      last=t;
    }
}
int len(struct Node *p)
{   int l=0;
    while(p)
    {   l++;
        p=p->next;
    }
    return l;
}
```

```
void Insert(struct Node *p, int index, int x)
{
    struct Node *t;
    int i;
    if(index < 0 || index > len(p))
    {
        return; }
    t=(struct Node *) malloc(sizeof(struct Node));
    t->data = x;
    if (index == 0)
    {   t->next = first;
        first = t;
    }
    else
    {  for(int i=0; i<index-1; i++)
            p = p->next;
        t->next = p->next;
        p->next = t;
    }
}

void Display(struct Node * p)
{
    if(p != NULL)
    {
        printf("%d", p->data);
        Display(p->next);
    }
}
```

```
int search(struct Node *p, int element)
{   int count=0;
    while(p!=NULL)
    {
        if(p->data==element)
        {   count++;
            p->data=-1;
        }
        p=p->next;
    }
    return count;
}

int main()
{
    Insert(first, 0, 15);
    Insert(first, 1, 16);
    Insert(first, 2, 14);
    while(first!=NULL)
    {
        if(first->data!=-1)
        {
            printf("%d comes %d times", first->data,
                                search(first, first->data));
        }
        first=first->next;
    }
}
```

DOMS Page No.
Date / /

Program - 6

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

/* create a node then linked list and length
   function and display function */
// Here, using function of program 5

void Taken(struct node *p, int n)
{   int i=0;
    struct Node * q=NULL;
    struct Node * t=p;
    struct Node * f=NULL, *temp=NULL;
    while (i<len(p)-n)
    {   q=p;
        p=p->next;
        temp=p;
    }
    while(p!=NULL)
    {   f=p;
        p=p->next;
    }
    f->next=t;
    q->next=NULL;
    first=temp;
}
void main()
{   int A[]={1,2,3,4,5,6,7}
    create(A,7);
    Taken(first, m);
    Display(first);
}
```

DOI 15 | Page No.
Date / /

Program - 7

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
} *first=NULL;
void create(int A[], int n)
{   int i;
    struct Node *t, *last;
    first=(struct Node*)malloc(sizeof(struct Node));
    first->data=A[0];
    first->next=NULL;
    last=first;
    for(i=1; i<n; i++)
    {   t=(struct Node*)malloc(sizeof(struct Node));
        t->data=A[i];
        t->next=NULL;
        last->next=t;
        last=t;
    }   }
int len(struct Node *p)
{   int count=0;
    while(p)
    {   count++;
        p=p->next;
    }
    return count;
}
```

```
void Display(struct Node *p)
{
    if(p!=NULL)
    {   printf("%d \t",p->data);
        Display(p->next);
    }
}
void Adding(struct Node *p)
{   struct Node *t=p;
    int number=0;
    int i=1,j=1000;
    while(t!=NULL)
    {   number+=i*(t->data);
        t=t->next;
        i=i/10;
    }
    number=number+1;
    while(p)
    {   p->data=(number/j);
        p=p->next;
        number=number%j;
        j=j/10;
    }
}
int main()
{   int A[]={1,9,9,9};
    create(A,4);
    Adding(first);
    Display(first);
}
```