# Institute of Engineering & Management
# Department of Computer Science & Engineering
# Data Structure Laboratory for 2nd year 3rd semester 2017
# Code: CS 392

**Date:** 23/8/17

## ASSIGNMENT-4

**Problem-1**

**Problem Statement:** Implement a simple linked list

**Algorithm:**

Step-1: START
Step-2: define a structure node type NODE with variables num as integer and NODE pointer ptr
Step-3: declare a NODE pointer head assigne NULL to it.
Step-4: create a function alloc(), inside alloc()
    return (NODE *)malloc(sizeof(NODE))
Step-5: Inside main(), declare NODE pointer temp and inp & flag=0 as integers
Step-6: do (repeat)
    print user commands & scan for inp
    switch for the value of inp between
        case 1: call insert_beg() & break
        case 2: call insert_end() & break
        case 3: call insert_any() & break
        case 4: call delete_beg() & break
        case 5: call delete_end() & break
        case 6: call delete_any() & break
        case 7: call rev_prnt() & break
        case 8: call search() & break
        case 9: print count_node() & break
        case 10: call split() & break
        case 11: call display() & break
        default: print “wrong input”
    print “enter 1 to continue” and scan for flag
  while flag = 1
Step-7: temp=head & repeat while head != NULL
    head = temp->ptr, free(temp) & temp = head
Step-8: Inside insert_beg(), declare NODE pointer temp=head
Step-9: head=alloc() & if head = NULL then print error & return
Step-10:head->ptr = temp & scan for head -> num
Step-11:Inside insert_end(), declare NODE pointer temp = head
Step-12:repeat while temp->ptr = NULL
    temp = temp->ptr
Step-13:temp -> ptr = alloc()
Step-14:if temp -> ptr = NULL, then print “error” and return
Step-15:temp -> ptr -> ptr = NULL
Step-16:scan for tem -> ptr -> num
Step-17:Inside insert_any(), declare NODE pointer temp1=head, temp2=head & new and integer variable pos
Step-18:print “enter position” & scan for pos

**Name:** Ranajit Roy, **Sec:** A, **Roll:** 47

Step-19:if pos = 0, then call insert_beg() and return
Step-20:repeat while pos != 0 and temp != NULL
pos = pos+1 & temp2 = temp1
temp1 = temp1 -> ptr
Step-21:new = alloc() & if new = NULL, then print “error” & return
Step-22:new -> ptr = temp1 & temp2 = new -> ptr
Step-23:print “enter data” & scan for new -> num
Step-24:Inside del_beg(), declare NODE pointer temp
Step-25:if head = NULL, then print “no nodes to delete” and return
Step-26:temp = head -> ptr & free(head)
Step-27:head = temp & print “deleted”
Step-28:Inside del_end(), declare NODE pointer temp = head, temp2 = head
Step-29:if head = NULL, then print “no nodes to delete” & return
Step-30:repeat while temp -> ptr != NULL
temp2 = temp & temp = temp -> ptr
Step-31:free(temp -> ptr), temp2 -> ptr = NULL & print “deleted”
Step-32:Inside del_any(), declare NODE pointer temp1 = head & temp2 = head
& integer variable elm;
Step-33:if head = NULL, then print “no nodes to delete” and return
Step-34:print “enter the fdata to delete” & scan for elm
Step-35:repeat while temp != NULL
if head -> num = elm
call del_beg(), temp1 = head & temp2 = head
else if temp1->num = elm
temp2 -> ptr = temp1 -> ptr & free(temp1)
temp1 = temp2 -> ptr & printf "deleted”
else temp2=temp1 & temp1=temp1->ptr
Step-36:Inside rev_prnt(), declare NODE pointer temp1 = head, temp2, temp3=NULL
Step-37:if head = NULL, then print “no elements to print” & return
Step-38:repeat while temp1 != NULL
temp1 = temp1 -> ptr & temp2 = head
repeat while temp2 -> ptr != temp3
temp2 = temp2 -> ptr
temp3 = temp2 & print temp2 -> num
Step-39:Inside search(), declare NODE pointer temp = head and integer variables
elm, flag = 0 & loc = 0
Step-40:if head = NULL, then print “no elements to search for” & return
Step-41:print “enter the elements to search” & scan for elm
Step-42:repeat while temp != NULL
if temp -> num = elm
flag = flag+1 &break
temp=temp->ptr & loc = loc+1
Step-43:if flag = 0, then print “no such record is found”
else print the position of data i.e. loc
Step-44:Inside count_node(), declare NODE pointer temp & integer variable count=0
Step-45:repeat while temp != NULL
temp = temp -> ptr & count = count+1
Step-46:return count
Step-47:Inside split(), declare NODE pointer head2 & temp = head and integer
variable n = count_node()/2
Step-48:if n = 0, then print “not enough elements to split” & return

Step-49:repeat while (n-1) != 0
temp = temp -> ptr & n = n-1
Step-50:head2 = temp -> ptr & temp -> ptr = NULL
Step-51:print “Elements in 1st linked list are ”
Step-52:temp = head & repeat while temp != NULL
print temp->num & temp = temp->ptr
Step-53:print “Elements in 2nd linked list are ”
Step-54: temp = head2 & repeat while temp != NULL
print temp->num & temp = temp->ptr
Step-55:Inside display(), declare NODE pointer temp = head
Step-56:if head = NULL, then print “no elements to display” & return
Step-57:repeat while temp != NULL
print temp -> num & temp = temp -> ptr

**Source code:**

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

typedef struct node
{
      int num;
      struct node *ptr;
} NODE;

NODE *head=NULL;

NODE *alloc()
{
      return (NODE *)malloc(sizeof(NODE));
}
void insert_beg();
void insert_end();
void insert_any();
void del_beg();
void del_end();
void del_any();
void rev_prnt();
void search();
int count_node();
void split();
void display();

void main()
{
      NODE *temp;
      int inp, flag=0;
      do
      {
            printf("Enter the following commands\n '1' to
                  insert at beginning\n '2' to insert
                  at end\n '3' to insert at specific
                  position\n '4' to delete from beginning\n
                  '5' to delete from end\n '6' to delete a
                  data\n '7' to reverse print\n '8' to
                  search\n '9' to count nodes\n '10' to split
                        in half\n '11' to display\n");
            scanf("%d", &inp);
```

```
            switch(inp)
            {
                  case 1:     insert_beg(); break;
                  case 2:     insert_end(); break;
                  case 3:     insert_any(); break;
                  case 4:     del_beg(); break;
                  case 5:     del_end(); break;
                  case 6:     del_any(); break;
                  case 7:     rev_prnt(); break;
                  case 8:     search(); break;
                  case 9:     printf("there are %d nodes in
                                    the linked list\n",
                                    count_node()); break;
                  case 10:    split(); break;
                  case 11:    display(); break;
                  default:    printf("wrong input");
            }
            printf("enter 1 to continue\n");
            scanf("%d", &flag);
      } while(flag==1);
      temp=head;
      while(head!=NULL)
      {
            head=temp->ptr;
            free(temp);
            temp=head;
      }
}

void insert_beg()
{
      NODE *temp=head;
      head=alloc();
      if(head==NULL)
      {
            printf("error\n"); return;
      }
      head->ptr=temp;
      printf("enter the data\n");
      scanf("%d", &head->num);
}

void insert_end()
{
      NODE *temp=head;
      while(temp->ptr!=NULL)
            temp=temp->ptr;
      temp->ptr=alloc();
      if(temp->ptr==NULL)
      {
            printf("error\n"); return;
      }

      temp=temp->ptr;
      temp->ptr=NULL;
      printf("Enter the data\n");
      scanf("%d", &temp->num);
}
```

```
void insert_any()
{
      NODE *temp1=head, *temp2=head, *new; int pos;
      printf("enter the position\n");
      scanf("%d", &pos);
      if(pos==0)
      {
            insert_beg(); return;
      }
      while(pos-- && temp1!=NULL)
      {
            temp2=temp1; temp1=temp1->ptr;
      }
      new=alloc();
      if(new==NULL)
      {
            printf("error\n"); return;
      }
      new->ptr=temp1;
      temp2->ptr=new;
      printf("enter the data\n");
      scanf("%d", &new->num);
}

void del_beg()
{
      NODE *temp;
      if(head==NULL)
      {
            printf("no nodes to delete\n"); return;
      }
      temp=head->ptr;
      free(head);
      head=temp;
      printf("deleted\n");
}

void del_end()
{
      NODE *temp=head, *temp2=head;
      if(head==NULL)
      {
            printf("no nodes to delete\n"); return;
      }
      while(temp->ptr!=NULL)
      {
            temp2=temp;
            temp=temp->ptr;
      }
      free(temp2->ptr);
      temp2->ptr=NULL;
      printf("deleted\n");
}

void del_any()
{
      NODE *temp1=head, *temp2=head; int elm;
      if(head==NULL)
```

```
	{
		printf("no nodes to delete\n"); return;
	}
	printf("enter the data to delete\n");
	scanf("%d", &elm);
	while(temp1!=NULL)
	{
		if(head->num==elm)
		{
			del_beg(); temp1=head; temp2=head;
		}
		else if(temp1->num==elm)
		{
			temp2->ptr=temp1->ptr; free(temp1);
			temp1=temp2->ptr; printf("deleted\n");
		} else {temp2=temp1;
		temp1=temp1->ptr;}
	}
}

void rev_prnt()
{
	NODE *temp1, *temp2, *temp3=NULL;
	temp1=head;
	if(head==NULL)
	{
		printf("no elements to print\n"); return;
	}
	while(temp1!=NULL)
	{
		temp1=temp1->ptr; temp2=head;
		while(temp2->ptr!=temp3)
			temp2=temp2->ptr;
		temp3=temp2;
		printf("%d, ", temp2->num);
	}
}

void search()
{
	NODE *temp=head; int elm, flag=0, loc=0;
	if(head==NULL)
	{
		printf("no elements to search for\n"); return;
	}
	printf("Enter the element to search\n");
	scanf("%d",&elm);
	while(temp!=NULL)
	{
		if(temp->num==elm)
		{
			flag++; break;
		}
		temp=temp->ptr;
		loc++;
	}
	if(flag==0)
		printf("no such record is found\n");
```

```
      else printf("the element '%d' is present in %d
                        location\n", elm, loc);
}

int count_node()
{
      NODE *temp=head; int count=0;
      while(temp!=NULL)
      {
            temp=temp->ptr;
            count++;
      }
      return count;
}

void split()
{
      int n=count_node()/2;
      NODE *head2, *temp=head;
      if(n==0)
      {
            printf("Not enough elements to split\n");
            return;
      }
      while(n-1)
      {
            temp=temp->ptr; n--;
      }
      head2=temp->ptr; temp->ptr=NULL;
      temp=head; printf("Elements in 1st linked list are ");
      while(temp!=NULL)
      {
            printf("%d, ", temp->num);
            temp=temp->ptr;
      }
      temp=head2;
      printf("\nElements in 2nd linked list are ");
      while(temp!=NULL)
      {
            printf("%d, ", temp->num);
            temp=temp->ptr;
      }
}

void display()
{
      NODE *temp=head;
      if(head==NULL)
      {
            printf("no elements to display\n"); return;
      }
      printf("Elements in the linked list are ");
      while(temp!=NULL)
      {
            printf("%d, ", temp->num);
            temp=temp->ptr;
      }
}
```

**Input/Output:**

```
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
1
enter the data
45
enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
2
Enter the data
34
enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
1
enter the data
89
enter 1 to continue
1
```

```
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
11
Elements in the linked list are 89, 45, 34, enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
3
enter the position
1
enter the data
70
enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
11
Elements in the linked list are 89, 70, 45, 34, enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
```

'2' to insert at end
'3' to insert at specific position
'4' to delete from beginning
'5' to delete from end
'6' to delete a data
'7' to reverse print
'8' to search
'9' to count nodes
'10' to split in half
'11' to display
6
enter the data to delete
45
deleted
enter 1 to continue
1
Enter the following commands
'1' to insert at beginning
'2' to insert at end
'3' to insert at specific position
'4' to delete from beginning
'5' to delete from end
'6' to delete a data
'7' to reverse print
'8' to search
'9' to count nodes
'10' to split in half
'11' to display
11
Elements in the linked list are 89, 70, 34, enter 1 to continue
1
Enter the following commands
'1' to insert at beginning
'2' to insert at end
'3' to insert at specific position
'4' to delete from beginning
'5' to delete from end
'6' to delete a data
'7' to reverse print
'8' to search
'9' to count nodes
'10' to split in half
'11' to display
8
Enter the element to search
70
the element '70' is present in 1 location
enter 1 to continue
1
Enter the following commands
'1' to insert at beginning

```
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
9
there are 3 nodes in the linked list
enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
7
34, 70, 89, enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
4
deleted
enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
```

```
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
5
deleted
enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
2
Enter the data
80
enter 1 to continue
1
Enter the following commands
 '1' to insert at beginning
 '2' to insert at end
 '3' to insert at specific position
 '4' to delete from beginning
 '5' to delete from end
 '6' to delete a data
 '7' to reverse print
 '8' to search
 '9' to count nodes
 '10' to split in half
 '11' to display
10
Elements in 1st linked list are 70,
Elements in 2nd linked list are 80, enter 1 to continue
0
```