Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

SSN College of Engineering

Department of Computer Science and Engineering

UCS2312 – Data Structures Lab

II Year CSE - B Section (III Semester)

Academic Year 2022-23

Staff Incharge : Dr.H.Shahul Hamead

Exercise 1 : Exercises on Single Linked List

Aim:

To implement basic C programs by using Single Linked List.

Basic :

Implement a single linked list abstract data type (ADT) with Create, print, search, insert (at the middle, head, end), delete (at the middle, head, end) functions :

Ex- no: 1 Name: M.Rohith
Date: 26-10-22 3122 21 5001 085

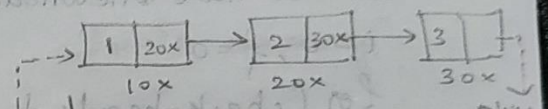
Pseudocode :

26.10.2022.

Exercises on singly linked list.

Basic:-

Implement a singly linked list abstract Data Type (ADT).



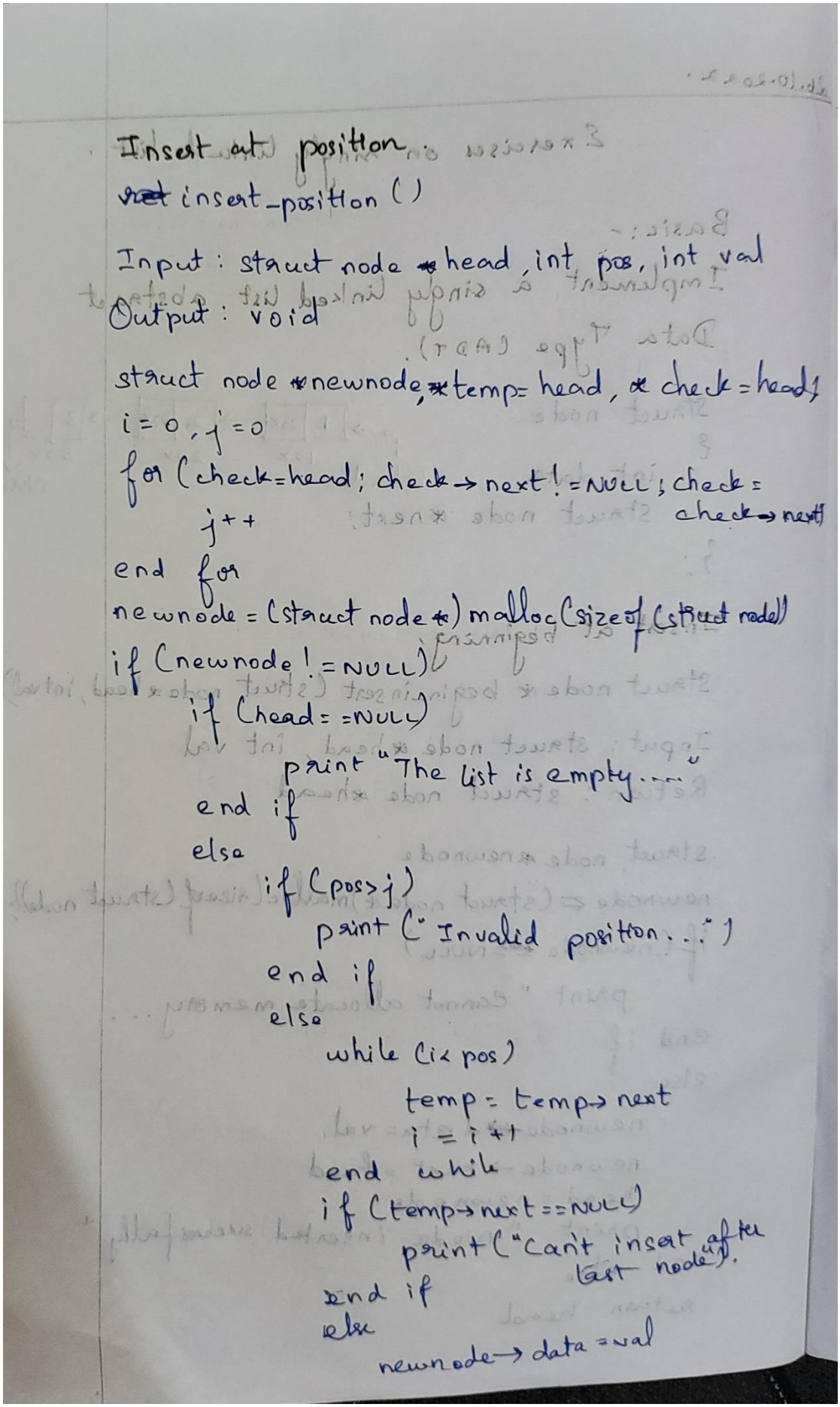

```
struct node
{
    int data
    struct node *next;
};
```

Insert at beginning:

```
struct node * begin_insert (struct node *head, int val)
Input : struct node *head, int val
Return : struct node *head

struct node *newnode
newnode = (struct node*)malloc(sizeof(struct node));
if (newnode == NULL)
        print "Cannot allocate memory....
end if
else
        newnode->data = val
        newnode->next = head
        head = newnode
        print "Node inserted successfully"
end else
return head
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085



Insert at position

insert_position()

Input : struct node *head, int pos, int val
Output : void

```
struct node *newnode, *temp= head, *check = head;
i = 0, j = 0
for (check=head; check -> next != NULL; check =
      j++                                check -> next)
end for
newnode = (struct node *) malloc (sizeof (struct node))
if (newnode != NULL)
    if (head == NULL)
        print "The list is empty...."
    end if
    else
        if (pos > j)
            print (" Invalid position...")
        end if
        else
            while (i < pos)
                temp = temp -> next
                i = i + 1
            end while
            if (temp -> next == NULL)
                print ("Can't insert after
                                last node").
            end if
            else
                newnode -> data = val
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

```
            newnode → next = temp → next
            temp → next = newnode
            print ("Node inserted successfully");
        end else
    end else
end if
else
    print " Cannot allocate memory"
end else
```

Last- insert().

```
last_insat()
Input : struct node *head, int val
Output : struct node *head
struct node *newnode, *temp = head
newnode = (struct node *) malloc (sizeof (struct node))
if (newnode == NULL)
    print " Cannot allocate new memory..."
end if
else
    while (temp → next != NULL)
        temp = temp → next
    end while
    newnode → data = val
    newnode → next = NULL
    temp → next = newnode.
    print (" Node inserted successfully"
end else
return head
```

Name: M.Rohith
3122 21 5001 085

Begin_delete.

```
begin_delete()
Input: struct node *head
Output: struct node *head

struct node *temp=head
if (temp==NULL)
    print "List is empty..."
    return head
end if
elseif (temp->next==NULL)
    free(temp)
    head=NULL
    print "Only node in the list is deleted..."
    return head
end else if
else
    head=temp->next;
    free(temp)
    print "Node deleted successfully..."
    return head
end else
```

Last delete.

```
last_delete()
Input: struct node *head
Output: struct node *head
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

```
struct node *temp1, *temp2
if (head == NULL)
    print "Nothing to delete..."
    return head
end if
else if (head->next == NULL)
    free (head)
    head = NULL
    print "Only one node in the list is
           deleted...."
    return head
end else if
else
    temp1 = head;
    while (temp1->next != NULL)
        temp2 = temp1
        temp1 = temp1->next
    end while
    free (temp1)
    print " Last node in the list is deleted.."
    temp2 -> next = NULL
    return head
end else
```

Delete at the position.

delete-position ()

Input: struct node *head, int pos

Output: void

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

```
struct node *check=head, *temp1, *temp2
int i=0
if (head==NULL)
    print "The list is empty...."
else
    while (check->next!=NULL)
        check=check->next
        i=i+1
    end while
    if (pos>i)
        print "Invalid position..."
    end if
    else if (pos==0)
        print "Cannot perform deletion at the
               begin...."
    end else if
    else if (pos==i)
        print "Cannot perform deletion at the
               last....."
    end else if
    else
        temp1=head
        i=0
        while(i!=pos)
            temp2=temp1
            temp1=temp1->next
            i=i+1
        end while
        temp2->next=temp1->next
        free(temp1)
        print(" Node deleted successfully...")
    end else
end else
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

Search.

```
Input: struct node *head, int val
Output: void
struct node *temp = head
int i=0, count=0
if (temp == NULL)
    print "List is empty"
end if
else
    if (temp->next == NULL)
        if (val == temp-> data)
            count = count+1
            print "Value is found at pos i"
        end if
    end if
    else
        while (temp != NULL)
            if (val == temp-> data)
                count = count+1
                print ("Value is found at pos i")
            end if
            i = i+1
            temp = temp-> next
        end while
    end else
    if (count == 0)
        print "The value is not found"
    end if
    else
        print "The value is found count times"
    end else
end else
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

Display.

```
display()
Input: struct node *head
Output: void
Struct node *ptr
ptr = head
if (ptr==NULL)
        print "Nothing to print...."
end if
else
        print "Displaying values...."
        while (ptr!=NULL)
                print "Memory= ptr \t Data = ptr->data
                        \t Next = ptr-> next"
                ptr = ptr -> next
        end while
        print "\n"
end else.
```

Main function.

```
main()
Variables: int choice=0, val=0, pos=0
           struct node *head=NULL
Output: void
```

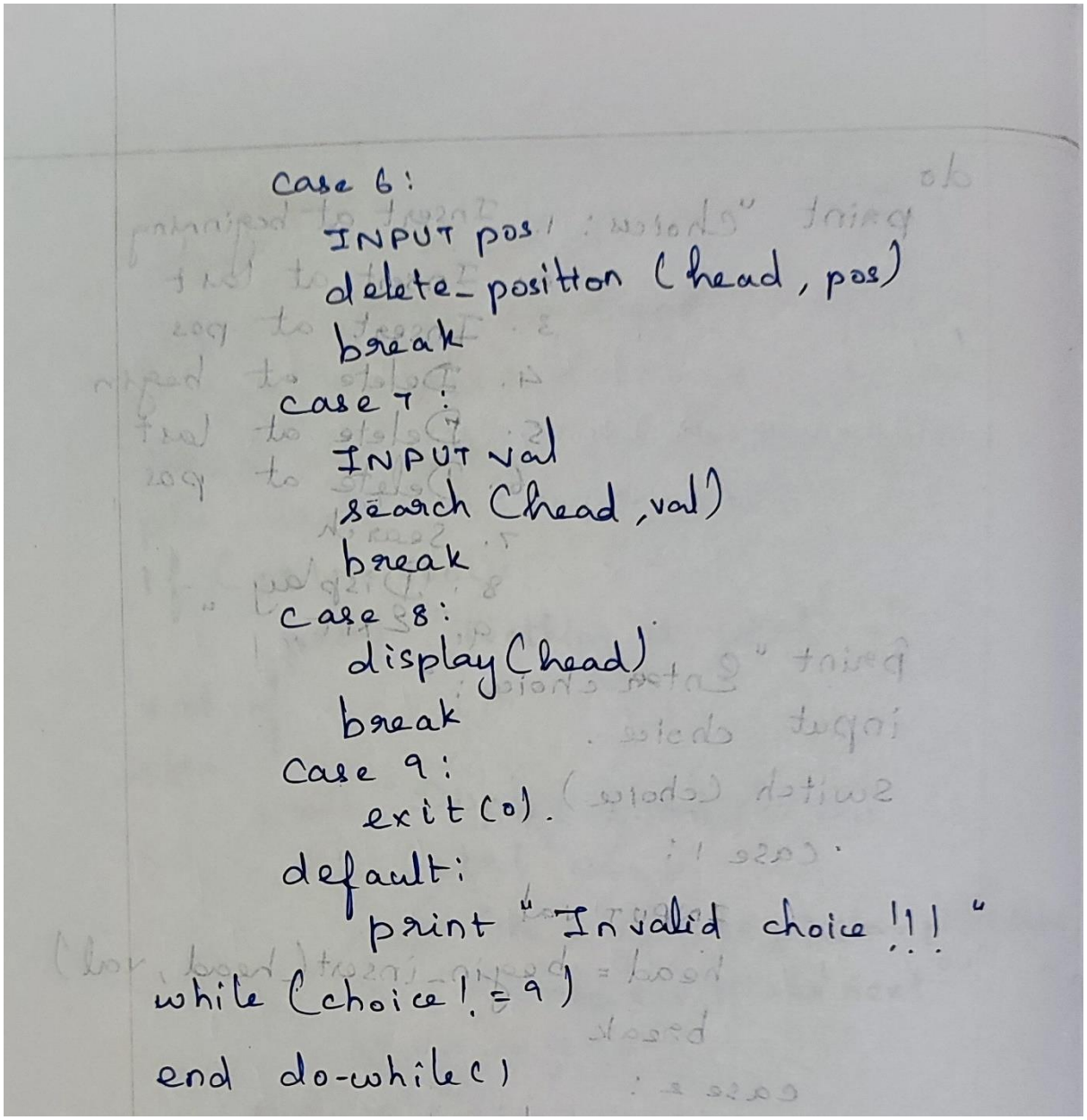
Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

```
do
  print "choices: 1. Insert at beginning
                  2. Insert at last
                  3. Insert at pos
                  4. Delete at begin
                  5. Delete at last
                  6. Delete at pos
                  7. Search
                  8. Display
                  9. Exit "
  print "Enter choice: "
  input choice.
  switch (choice)
     case 1:
        INPUT val
        head = begin_insert(head, val)
        break
     case 2:
        INPUT val
        head = last_insert(head, val)
        break
     case 3:
        INPUT pos
        INPUT val
        insert_position(head, pos, val)
        break
     case 4:
        head = begin_delete(head)
        break
     case 5:
        head= last_delete(head)
        break
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085



```
        case 6:
            INPUT pos
            delete_position (head, pos)
            break
        case 7:
            INPUT val
            search (head, val)
            break
        case 8:
            display (head)
            break
        case 9:
            exit (0).
        default:
            print " Invalid choice !!! "
while (choice != 9)
end do-while()
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

Program code:

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

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

struct node* begin_insert(struct node *head,int val)
{
    struct node *newnode;
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("\nCannot allocate new memory...\n");
    }
    else
    {
        newnode->data=val;
        newnode->next=head;
        head=newnode;
        printf("\nNode inserted successfully\n");
    }
    return head;
}

void insert_position(struct node *head,int pos,int val)
{
    struct node *newnode,*temp=head,*check=head;
    int i=0,j=0;
    for(check=head;check->next!=NULL;check=check->next)
        j++;
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode!=NULL)
    {
        if(head==NULL)
            printf("\nThe list is empty...\n");
        else
        {
            if(pos>j)
                printf("\nInvalid position...\n");
            else
            {
                while(i<pos)
                {
                    temp=temp->next;
```

Name: M.Rohith
3122 21 5001 085

```
                    i++;
                }
                if(temp->next==NULL)
                    printf("\nCan't insert after the last
node...\n");
                else
                {
                    newnode->data=val;
                    newnode->next=temp->next;
                    temp->next=newnode;
                    printf("\nNode inserted successfully\n");
                }
            }
        }
    }
    else
        printf("\nCannot allocate memory...\n");
}

struct node* last_insert(struct node* head,int val)
{
    struct node *newnode,*temp=head;
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("\nCannot allocate new memory...\n");
    }
    else
    {
        while(temp->next!=NULL)
            temp=temp->next;
        newnode->data=val;
        newnode->next=NULL;
        temp->next=newnode;
        printf("\nNode inserted successfully\n");
    }
    return head;
}

struct node* begin_delete(struct node* head)
{
    struct node *temp=head;
    if(temp==NULL)
    {
        printf("\nList is empty...\n");
        return head;
    }
    else if(temp->next==NULL)
```

Name: M.Rohith
3122 21 5001 085

```
    {
        free(temp);
        head=NULL;
        printf("\nOnly node in the list is deleted...\n");
        return head;
    }
    else
    {
        head=temp->next;
        free(temp);
        printf("\nNode deleted successfully...\n");
        return head;
    }
}

struct node* last_delete(struct node *head)
{
    struct node *temp1,*temp2;
    if(head==NULL)
    {
        printf("\nNothing to delete...\n");
        return head;
    }
    else if(head->next==NULL)
    {
        free(head);
        head=NULL;
        printf("\nOnly one node in the list is deleted...\n");
        return head;
    }
    else
    {
        temp1=head;
        while(temp1->next!=NULL)
        {
            temp2=temp1;
            temp1=temp1->next;
        }
        free(temp1);
        printf("\nLast node in the list is deleted...\n");
        temp2->next=NULL;
        return head;
    }
}

void delete_position(struct node *head,int pos)
{
    struct node *check=head,*temp1,*temp2;
```

Name: M.Rohith
3122 21 5001 085

```
    int i=0;
    if(head==NULL)
        printf("\nThe list is empty...\n");
    else
    {
        while(check->next!=NULL)
        {
            check=check->next;
            i++;
        }
        if(pos>i)
            printf("\nInvalid position...\n");
        else if(pos==0)
            printf("\nCannot perform deletion at the begin...\n");
        else if(pos==i)
            printf("\nCannot perform deletion at the last...\n");
        else
        {
            temp1=head;
            i=0;
            while(i!=pos)
            {
                temp2=temp1;
                temp1=temp1->next;
                i++;
            }
            temp2->next=temp1->next;
            free(temp1);
            printf("\nNode deleted successfully...\n");
        }
    }
}

void search(struct node *head,int val)
{
    struct node *temp=head;
    int i=0,count=0;
    if(temp==NULL)
        printf("\nList is Empty");
    else
    {
        if(temp->next==NULL)
        {
            if(val==temp->data)
            {
                count++;
                printf("\nValue %d is found at the position %d",val,i);
            }
```

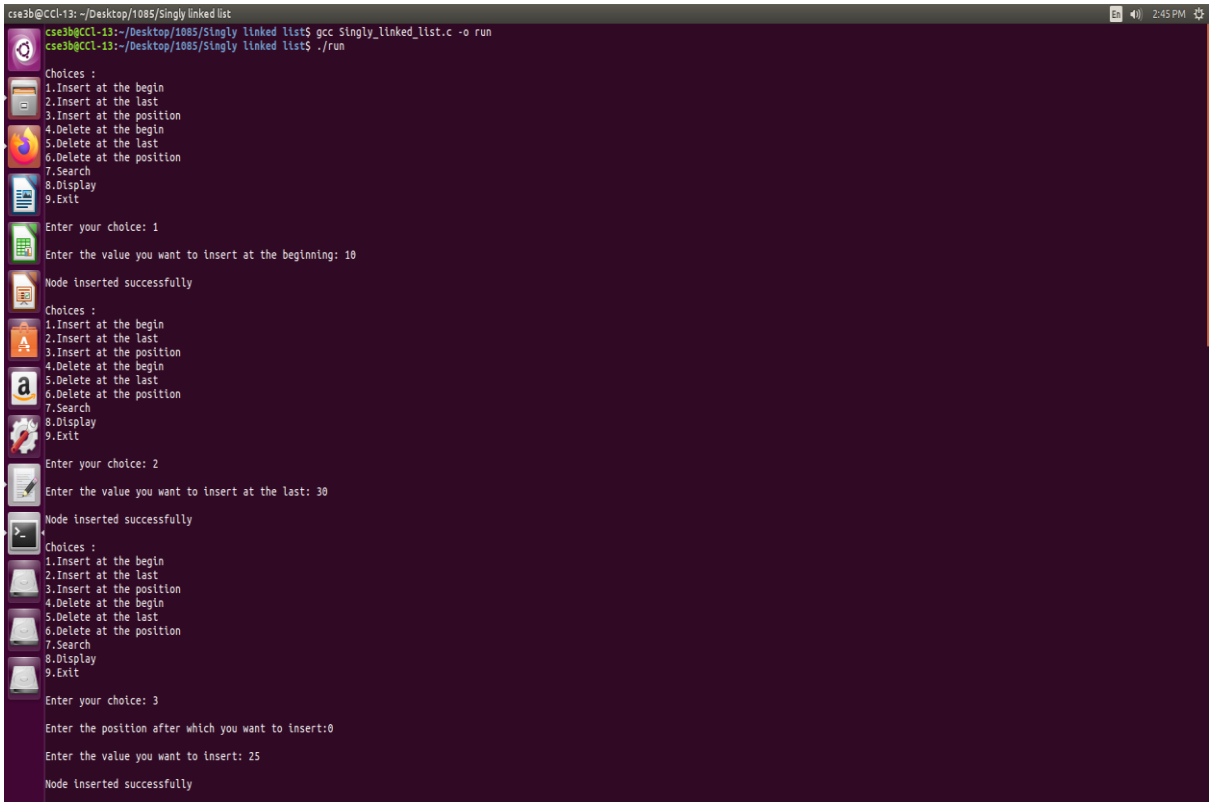
```
        }
        else
        {
            while(temp!=NULL)
            {
                if(val==temp->data)
                {
                    count++;
                    printf("\nValue %d is found at the position %d",val,i);
                }
                i++;
                temp=temp->next;
            }
        }
        if(count==0)
            printf("\nThe Value %d is not found in the List",val);
        else
            printf("\nThe Value is found %d time(s)",count);
    }
}

void display(struct node *head)
{
    struct node *ptr;
    ptr=head;
    if(ptr==NULL)
    {
        printf("\nNothing to print...\n");
    }
    else
    {
        printf("\nDisplaying values...");
        while(ptr!=NULL)
        {
            printf("\nMemory=%p\tData=%d\tNext=%p",ptr,ptr->data,ptr->next);
            ptr=ptr->next;
        }
        printf("\n");
    }
}

void main()
{
    int choice=0,val=0,pos=0;
    struct node *head=NULL;
    do
    {
```

Name: M.Rohith
3122 21 5001 085

```
        printf("\nChoices :\n1.Insert at the begin\n2.Insert at the
last\n3.Insert at the position\n4.Delete at the begin\n5.Delete at the
last\n6.Delete at the position\n7.Search\n8.Display\n9.Exit\n");
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch (choice)
        {
            case 1:
                printf("\nEnter the value you want to insert at the beginning:
");
                scanf("%d",&val);
                head=begin_insert(head,val);
                break;
            case 2:
                printf("\nEnter the value you want to insert at the last: ");
                scanf("%d",&val);
                head=last_insert(head,val);
                break;
            case 3:
                printf("\nEnter the position after which you want to
insert:");
                scanf("%d",&pos);
                printf("\nEnter the value you want to insert: ");
                scanf("%d",&val);
                insert_position(head,pos,val);
                break;
            case 4:
                head=begin_delete(head);
                break;
            case 5:
                head=last_delete(head);
                break;
            case 6:
                printf("\nEnter the position you want to delete:");
                scanf("%d",&pos);
                delete_position(head,pos);
                break;
            case 7:
                printf("\nEnter the element to search: ");
                scanf("%d",&val);
                search(head,val);
                break;
            case 8:
                display(head);
                break;
            case 9:
                exit(0);
            default:
```

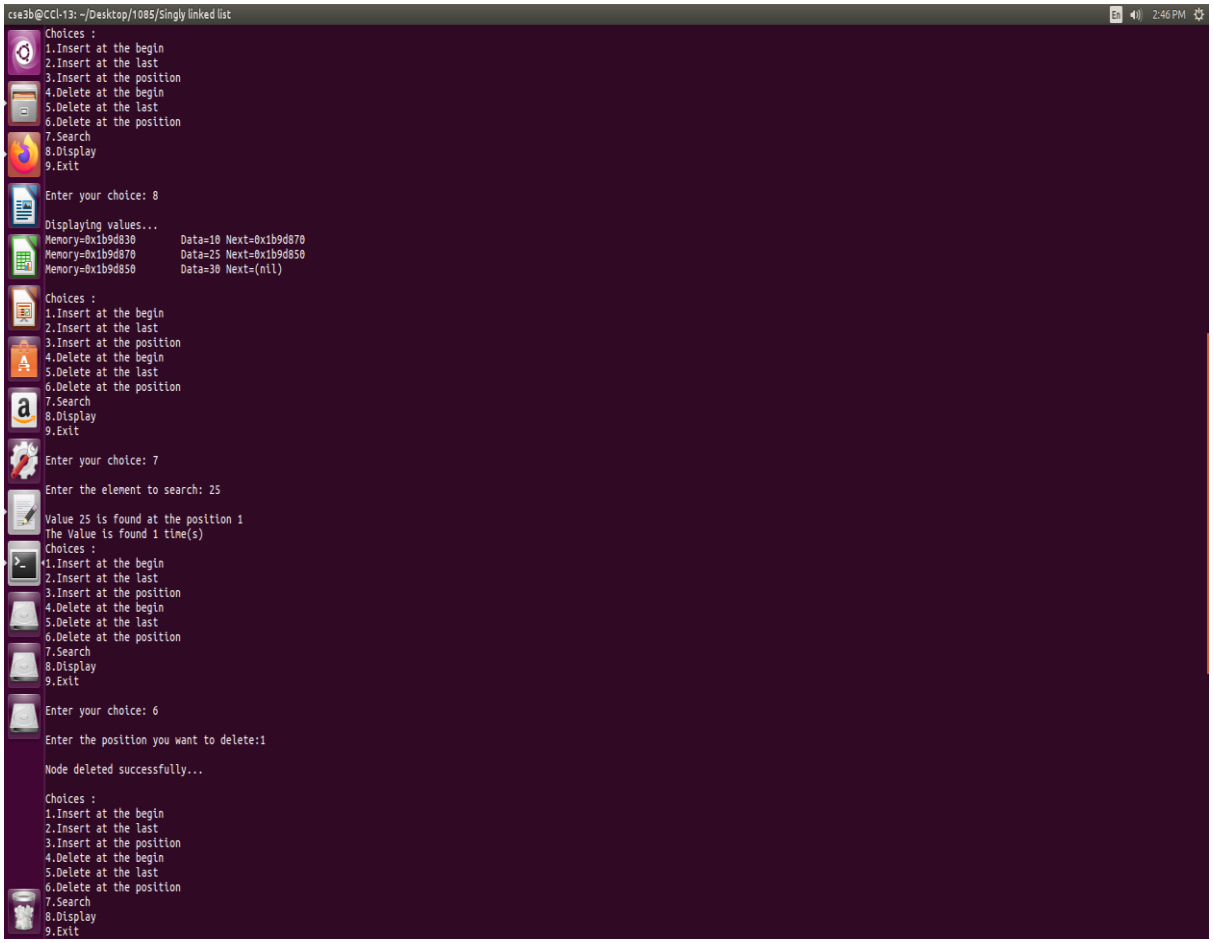
```
                printf("\nInvalid choice!!!");
        }
    }while(choice!=9);
}
```

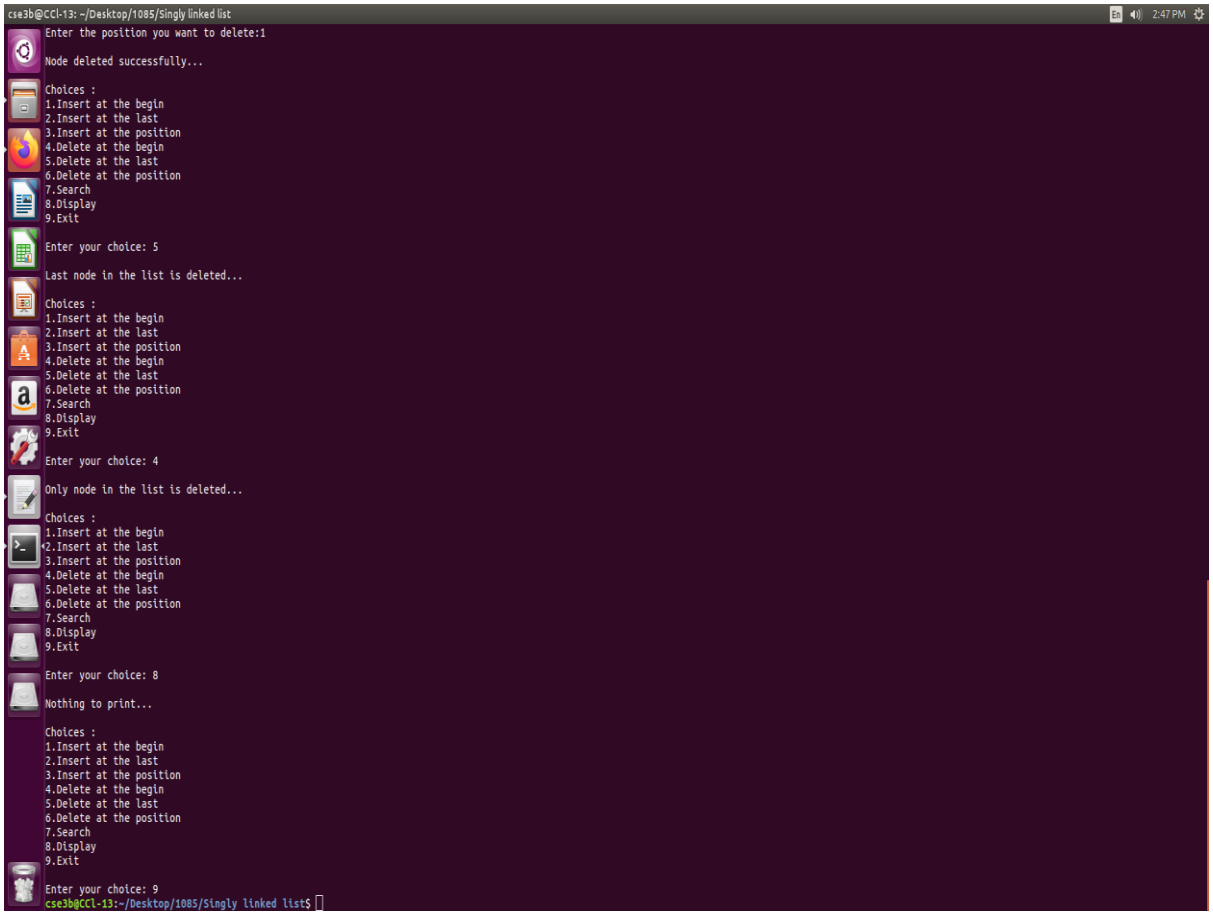
Output:



Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085





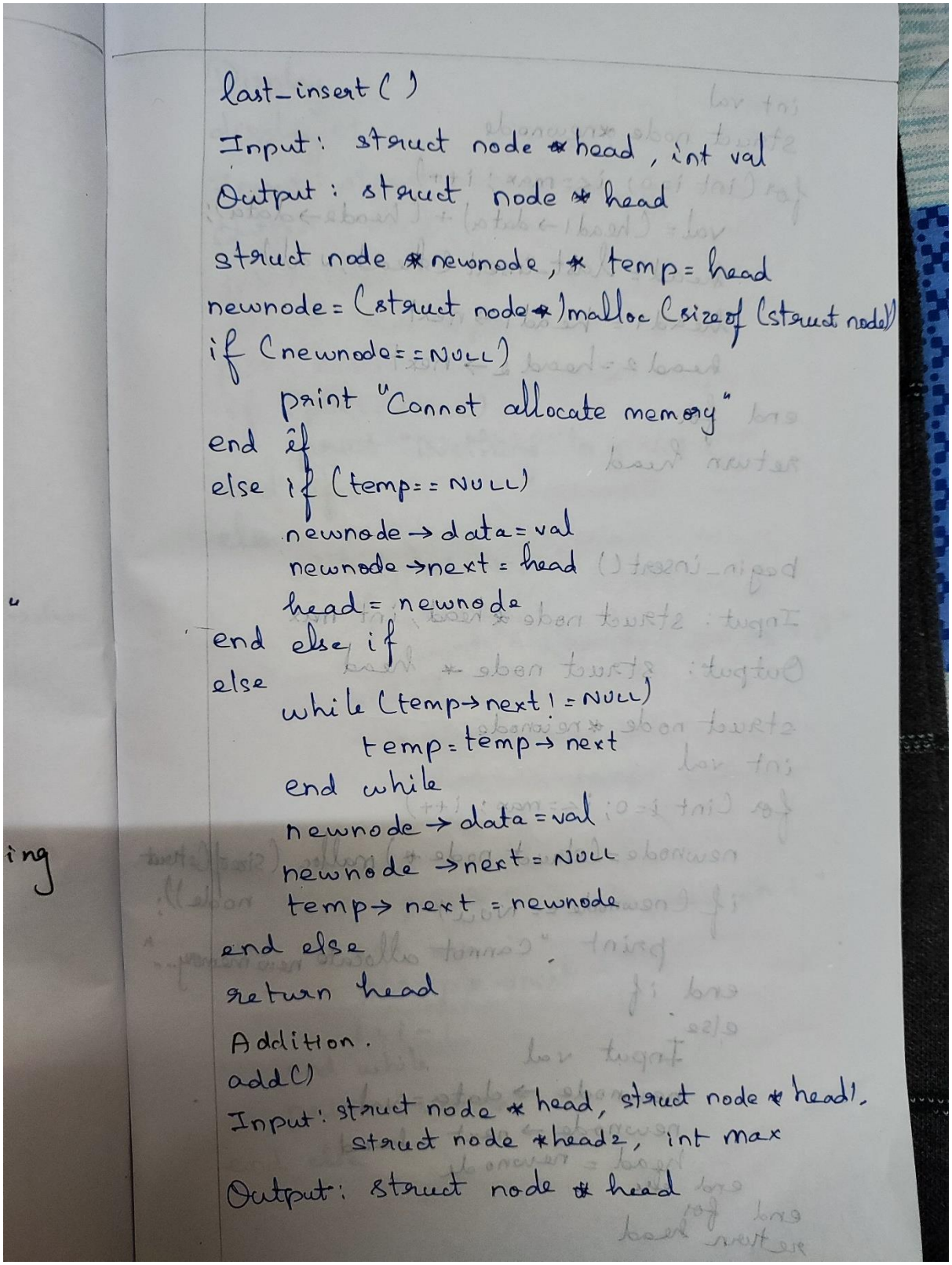
Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

Application:

Implement polynomial addition using single linked list ADT:

Pseudocode:



```
last_insert ( )
Input: struct node *head, int val
Output: struct node * head
struct node *newnode, * temp = head
newnode = (struct node *)malloc (sizeof (struct node))
if (newnode == NULL)
    print "Cannot allocate memory"
end if
else if (temp == NULL)
    newnode -> data = val
    newnode -> next = head
    head = newnode
end else if
else
    while (temp -> next != NULL)
        temp = temp -> next
    end while
    newnode -> data = val
    newnode -> next = NULL
    temp -> next = newnode
end else
return head

Addition.
add()
Input: struct node * head, struct node * head1,
       struct node *head2, int max
Output: struct node * head
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

```
int val
struct node *newnode
for(int i=0; i<=max; i++)
    val = (head1->data) + (head2->data);
    head = last_insert(head, val)
    head1 = head1->next
    head2 = head2->next
end for
return head
```

begin_insert()

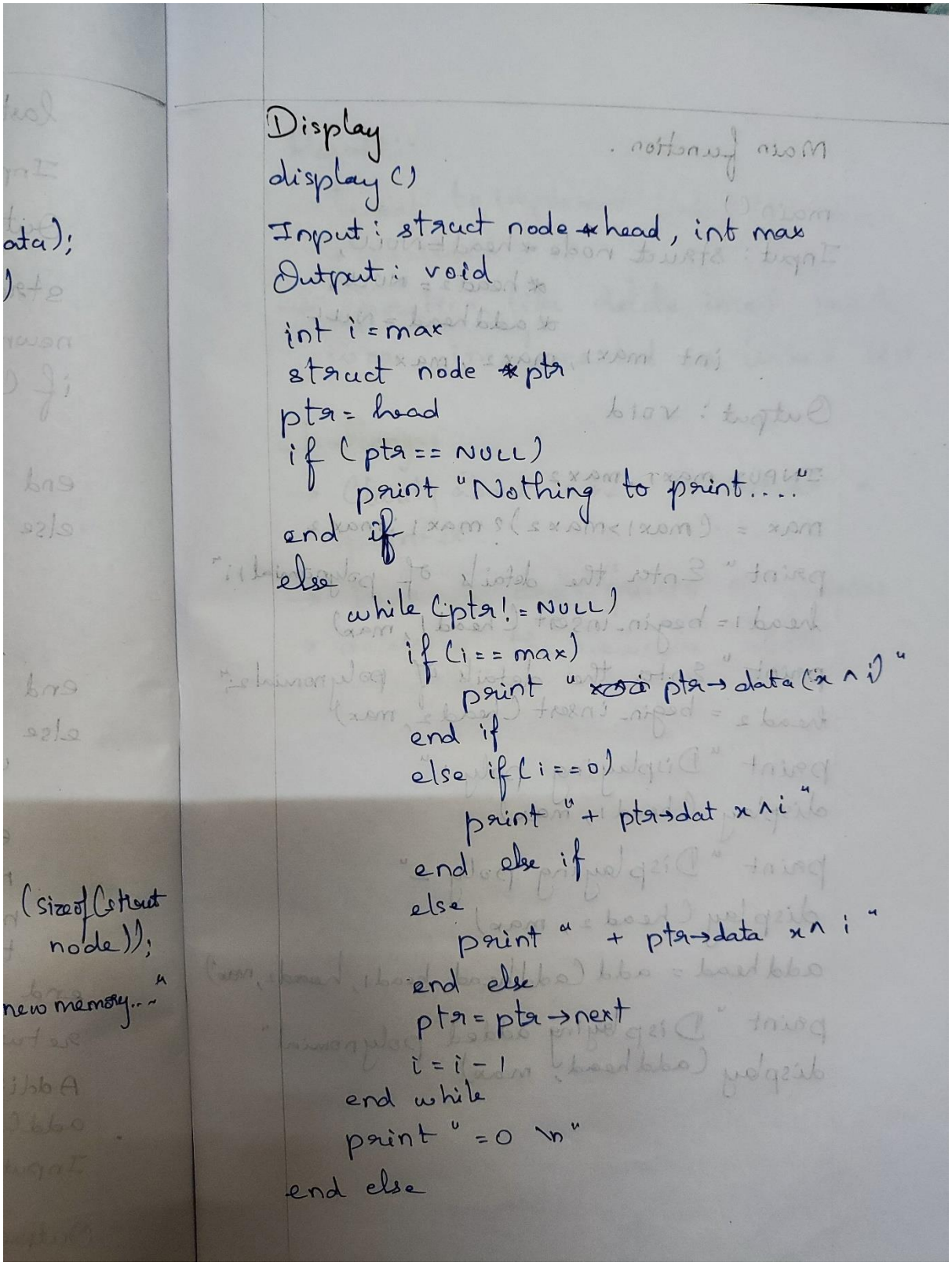
Input: struct node *head, int max

Output: struct node * head

```
struct node *newnode
int val
for(int i=0; i<=max; i++)
    newnode=(struct node *) malloc (sizeof(struct node));
    if(newnode == NULL)
        print "Cannot allocate new memory..."
    end if
    else
        Input val
        newnode -> data = val
        newnode -> next = head
        head = newnode
    end else
end for
return head
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085



```
Display
display()
Input: struct node *head, int max
Output: void
int i=max
struct node *ptr
ptr=head
if (ptr==NULL)
    print "Nothing to print...."
end if
else
    while(ptr!=NULL)
        if(i==max)
            print " ptr->data(x^i) "
        end if
        else if(i==0)
            print " + ptr->dat x^i "
        end else if
        else
            print " + ptr->data x^i "
        end else
        ptr=ptr->next
        i=i-1
    end while
    print " =0 \n"
end else
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

Main function.

main()
Input: struct node *head = NULL,
*head2 = NULL,
*addhead = NULL
int max1, max2, max

Output: void

INPUT max1, max2
max = (max1 > max2) ? max1 : max2
print " Enter the details of polynomial1"
head1 = begin_insert (head1, max)
print " Enter the details of polynomial2"
head2 = begin_insert (head2, max)
print " Displaying poly 1"
display (head1, max)
print " Displaying poly 2"
display (head2, max)
addhead = add (addhead, head1, head2, max)
print " Displaying added polynomial"
display (addhead, max)

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

Program code:

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

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

struct node* last_insert(struct node* head,int val)
{
    struct node *newnode,*temp=head;
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("\nCannot allocate new memory...\n");
    }
    else if(temp==NULL)
    {
        newnode->data=val;
        newnode->next=head;
        head=newnode;
    }
    else
    {
        while(temp->next!=NULL)
            temp=temp->next;
        newnode->data=val;
        newnode->next=NULL;
        temp->next=newnode;
    }
    return head;
}

struct node* add(struct node* head,struct node* head1,struct node* head2,int
max)
{
    int val;
    struct node* newnode;
    for(int i=0;i<=max;i++)
    {
        val=(head1->data)+(head2->data);
        head=last_insert(head,val);
        head1=head1->next;
        head2=head2->next;
    }
    return head;
```

Name: M.Rohith
3122 21 5001 085

```
}

struct node* begin_insert(struct node *head,int max)
{
    struct node *newnode;
    int val;
    for(int i=0;i<=max;i++)
    {
        newnode=(struct node*)malloc(sizeof(struct node));
        if(newnode==NULL)
        {
            printf("\nCannot allocate new memory...\n");
        }
        else
        {
            printf("\nEnter the coefficient of x^%d : ",i);
            scanf("%d",&val);
            newnode->data=val;
            newnode->next=head;
            head=newnode;
        }
    }
    return head;
}

void display(struct node *head,int max)
{
    int i=max;
    struct node *ptr;
    ptr=head;
    if(ptr==NULL)
    {
        printf("\nNothing to print...\n");
    }
    else
    {
        while(ptr!=NULL)
        {
            if(i==max)
                printf("%dx^%d",ptr->data,i);
            else if(i==0)
                printf("+ %dx^%d",ptr->data,i);
            else
                printf(" + %dx^%d",ptr->data,i);
            ptr=ptr->next;
            i--;
        }
        printf(" =0 \n");
```

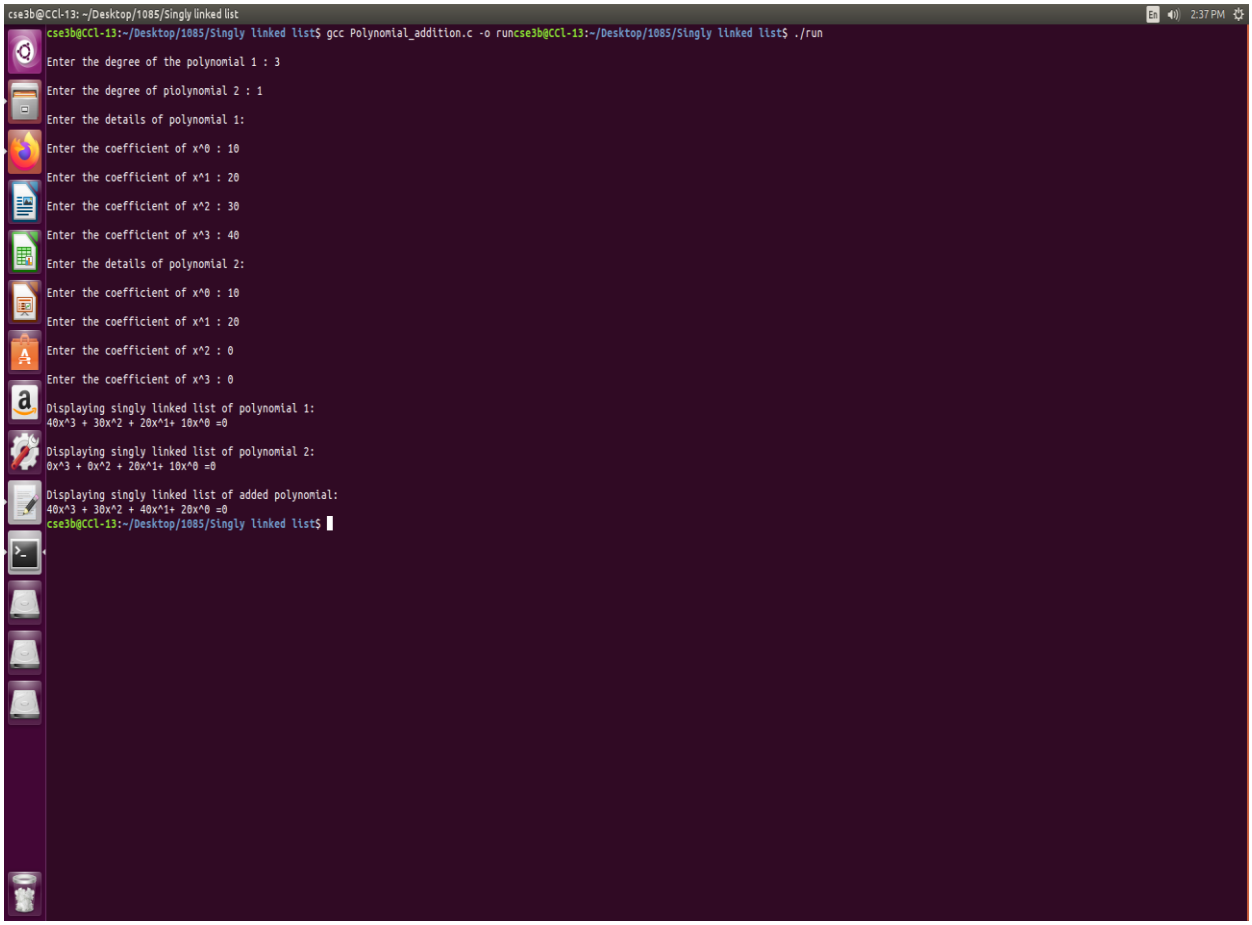
Name: M.Rohith
3122 21 5001 085

```
    }
}

void main()
{
    struct node *head1=NULL,*head2=NULL,*addhead=NULL;
    int max1,max2,max;
    printf("\nEnter the degree of the polynomial 1 : ");
    scanf("%d",&max1);
    printf("\nEnter the degree of piolynomial 2 : ");
    scanf("%d",&max2);
    max=(max1>max2)?max1:max2;

    printf("\nEnter the details of polynomial 1:\n");
    head1=begin_insert(head1,max);

    printf("\nEnter the details of polynomial 2:\n");
    head2=begin_insert(head2,max);
    printf("\nDisplaying singly linked list of polynomial 1:\n");
    display(head1,max);

    printf("\nDisplaying singly linked list of polynomial 2:\n");
    display(head2,max);

    addhead=add(addhead,head1,head2,max);
    printf("\nDisplaying singly linked list of added polynomial:\n");
    display(addhead,max);
}
```

Ex- no: 1
Date: 26-10-22

Name: M.Rohith
3122 21 5001 085

Output:



Results:

Thus C program using Single linked list in data structures as well as polynomial addition using singly linked list have been written and implemented successfully.