G. I. E. T. UNIVERSITY

GUNUPUR, Dist-Rayagada(O)

CONTENTS

LAB EXPERIMENT-1

(1) WAP to create methods for performing addition, substraction, multiplicartion and devision on 2 integers.

```
#include<stdio.h>
int sum(int a, int b)
{
 int c;
 c=a+b;
printf("sum of %d and %d is : %d \n", a, b, c);
}
int sub(int a, int b)
{
int c;
c=a-b;
printf("sub of %d and %d is : %d\n", a,b, c);
}
int mul(int a, int b)
{
 int c;
 c=a*b;
printf("mul of %d and %d is : %d\n", a, b, c);
}
int div(int a, int b)
{
int c;
 c=a/b;
```

```
printf("div of %d and %d is : %d\n",a,b,c);
}
int main()
{
int a,b,c;
printf("enter two integer no:");
scanf("%d%d",&a,&b);
sum(a,b);
sub(a,b);
mul(a,b);
div(a,b);
return 0;
}
```

o/p → Enter two integer no: 4 2
sum of 4 and 2 is: 6
sub of 4 and 2 is: 2
Mul of 4 and 2 is: 8
div of 4 and 2 is: 2

② WAP to create UDF for input 10 numbers into a 1D array create two functions max() and min(). max() is used to return the largest element and min is used to return the smallest number in array.

```
#include<stdio.h>
int max(int a[], int n)
{
int max, i;
max = a[0];
for(i=1; i<n; i++)
{
if(max<a[i])
max = a[i]
}
printf("maximum of array is : %d\n", max);
int min(int a[], int n)
{
int min, i;
min = a[0];
for(i=1; i<n; i++)
{
if(min>a[i])
min = a[i];
}
printf("minimum of array is : %d\n", min);
```

```
int main()
{
int a[20],i,n,sum;
printf("enter the size of array:");
scanf("%d",&n);
printf("\n enter elements of array:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
max(a,n);
min(a,n);
return 0;
}
```

o/p → Enter the size of array : 10
Enter the element of array :
11
22
33
40
42
45
46
47
50
51

Maximum of array is : 51
Minimum of array is : 11

(3) WAP a 'c' program to create methods for operation insertion and display on 1D array of elements using UDF.

```
#include<stdio.h>
int display(int a[], int n)
{
int i;
printf("the array after inserting the new element is \n");
for(i=0;i<n;i++)
  {
   printf("a[%d]:[%d]\n", i, a[i]);
  }
}
int insert(int a[], int n)
{
int pos, i, ITEM;
printf("Enter the position where you want to insert element");
scanf("%d", &pos);
printf("Enter the Item into the position");
scanf("%d", &ITEM);
for(i=n-1; i>pos-1; i--)
  { a[i+1] =a[i];
  }
```

```
a[pos-1] = ITEM;
n = n+1;
display(a,n);
}
int main()
{
int a[40], pos, i, n, ITEM;
printf("Enter no. of elements in array:");
scanf("%d", &n);
printf("Enter %d element:\n", n);
for(i=0; i<n; i++)
{
scanf("%d", &a[i]);
}
for(i=0; i<n; i++)
{
printf("a[%d]: %d \n", i, a[i]);
}
insert(a,n);
return 0;
}
```

14/04/23

o/p → Enter the no. of element in array: 6
Enter 6 element in array:
20
21
22
23
27
28

20 21 22 23 27 28

Enter the position where you want to insert: 5
Enter the item into that position: 301
the array after inserting the new element:

a[0] = 20
a[1] = 21
a[2] = 22
a[3] = 23
a[4] = 24
a[5] = 27
a[6] = 28

④ WAP to create method for operation deletion and display on 1d array of the element using UDF.

```
#include<stdio.h>
int display(int a[], int n)
{
int i;
printf("\n the resultant of array is :\n");
for(i=0;i<n;i++)
 {
    printf("a[%d]: %d \n",i,a[i]);
 }
}
int delet(int a[], int n)
 {
  int pos,i;
printf("Define the position of the array element whe
                                    you want to delet:\n");
scanf("%d",&pos);
for(i=pos-1;i<n-1;i++)
 {a[i]=a[i+1]
 }
 n=n-1;
}display(a,n);
```

```
int main()
{
int a[40], pos, i, n;
printf("Enter no. of elements in array");
scanf("%d", &n);
printf("Enter %d element: \n", n);
for(i=0; i<n; i++)
{
scanf("%d", &a[i]);
}
for(i=0; i<n; i++)
{
printf("a[%d]: %d\n", i, a[i]);
}
delet(a,n);
return 0;
}
```

O/p → Enter no. of elements in array : 5

Enter 5 element : 15
18
14
12
11

a[0] = 15
a[1] = 18
a[2] = 14
a[3] = 12
a[4] = 11

Define the position of the array element where you want to delet : 4

the resultant of array is :

a[0] = 15
a[1] = 18
a[2] = 14
a[3] = 11

LAB EXPERIMENT -2

① WAP to create function for performing matrix multiplication using UDF.

```
#include<stdio.h>
int mult(int r1, int c1, int r2, int c2)
{ int i,j,k;
  int a[10][10],b[10][10],mul[10][10];
if(c1==r2)
{
printf("enter the elements of the first matrix:\n");
for(i=0;i<r1;i++)
{ for(j=0;j<c1;j++)
  { printf("a[%d][%d]=",i,j);
    scanf("%d",&a[i][j]);
  }
}
printf("enter the elements of the second matrix:\n");
  for(i=0;i<r2;i++)
  { for(j=0;j<c2;j++)
    { printf("B[%d][%d]=",i,j);
      scanf("%d",&b[i][j]);
  } }
}
```

```
for(i=0; i<r1; i++)
  {for(j=0; j<c2; j++)
   {printf
    mul[i][j]=0;
   for(k=0; k<r2; k++)
    {mul[i][j]=mul[i][j]+a[i][k]*b[k][j];
    }
  }
}
printf("the multiplied matrix is :\n");
 for(i=0; i<r1; i++)
   {for(j=0; j<c2; j++)
     {printf("%d\t", mul[i][j]);
     }
    printf("\n");
   }
 }
else
  {printf("dimension does not match for multiplication
  }
}
```

```
int main()
{
    int i, j, r1, c1, r2, c2;
    printf("Enter row & column of first matrix:\n");
    scanf("%d %d", &r1, &c1);
    printf("Enter row & column of second matrix:\n");
    scanf("%d %d", &r2, c2);
    mult(r1, c1, r2, c2);
    return 0;
}
```

O/P → Enter row and coleemn of first matrix : 2 2
2
2
Enter row and coleemn of second matrix!
2
2
Enter the elements of the first matrix!
a[0][0] = 1
a[0][1] = 2
a[1][0] = 3
a[1][1] = 4
Enter the elements of the second matrix :
b[0][0] = 5
b[0][1] = 6
b[1][0] = 7
b[1][1] = 8
the multiplied matrix is :
19 22
43 50

② WAP to input elements into square matrix and display the transpose of it using UDF.

```
#include<stdio.h>
int display(int b[10][10], int r, int c)
{
 int i,j;
 printf("\n\n the transpose of a matrix is : ");
 for(i=0;i<c;i++)
  {
   printf("\n");
   for(j=0;j<r;j++)
    {
     printf("%d\t",b[i][j]);
    }
  }
 printf("\n\n");
}
int transpose(int a[10][10], int r, int c)
{
 int b[10][10],i,j;
 for(i=0;i<r;i++)
  {for(j=0;j<c;j++)
    {
     b[i][j]=a[i][j];
    }
  }
 display(b,r,c)
}
```

```
int main()
{
 int a[10][10], i, j, r, c;
 printf("\n\n transpose of a Matrix:\n");
 printf("----------\n");
 printf("\n input the rows & columns of the matrix");
 scanf("%d %d", &r, &c);
 printf("\n input elements in the first matrix:\n");
for(i=0; i<r; i++)
 {for(j=0; j<c; j++)
    {
     printf("element-[%d][%d]: ", i, j);
     scanf("%d", &a[i][j]);
    }
 }
printf("\n the matrix is:\n");
 for(i=0; i<r; i++)
 {
  printf("\n");
  for(j=0; j<c; j++)
     printf("%d\t", a[i][j]);

 }
 transpose(a, r, c);
 return 0;

}
```

Q/1 → transpose of a matrix:

Input the row and coleumn of the matrix
2
Input elements in the first matrix:
Element - [0][0] - 1
Element - [0][1] - 2
Element - [1][0] - 3
Element - [1][1] - 4
the matrix is:
1 2
3 4
the transpose of a matrix is:
1 3
2 4

(8) WAP in C to input in a 4x4 & display sum for individual row element using UDF.

```
#include<stdio.h>
int rowsum(int a[10][10], int M, int n)
{
  int i,j, sum=0;
  for(i=0; i<M; i++)
  {for(j=0; j<n; j++)
    {sum=sum+a[i][j];
    }
    printf("sum of the %d row= %d\n", i, sum);
    sum=0;
  }
}
int colsum(int a[10][10], int M, int n)
{
  int i, j, sum=0;
  for(j=0; j<n; j++)
  {
    for(i=0; i<M; i++)
    {
      sum=sum+a[i][j];
    }
    printf("sum of %d column= %d\n", j, sum);
  } sum=0;
}
```

```
int main()
{
 int a[10][10], i, j, m, n;
 printf("Enter the row and column size of the matrix\n");
 scanf("%d %d", &m, &n);
 for(i=0; i<m; i++)
  {
   for(j=0; j<n; j++)
    {
     printf("a[%d][%d]=", i, j);
     scanf("%d", &a[i][j]);
    }
  }
 rowsum(a, m, n);
 colsum(a, m, n);
 return 0;
}
```

o/p → Enter the row & column size of the matrix

2

2

a[0][0] = 1

a[0][1] = 2

a[1][0] = 3

a[1][1] = 4

sum of the 0 row is = 3

sum of the 1 row is = 7

sum of 0 column is = 4

sum of 1 column is = 6

(4) write a program to input elements into a 4x4 matrix, check it for sparse or not. If sparse then store the non-zero elements into an alternate matrix and then display it using UDF.

```
#include<stdio.h>
int a,b,i,j,k,zero=0,nonzero=0,arr[10][10];
int main()
{
printf("Enter the no. of rows and columns of the matrix:");
scanf("%d%d",&a,&b);
printf("Enter %d elements of the matrix:",a*b);
for(i=0;i<a;i++)
{for(j=0;j<b;j++)
  {scanf("%d",&arr[i][j]);
  }
}
printf("the matrix entered is:\n");
for(i=0;i<a;i++)
 {for(j=0;j<b;j++)
   {printf("[%d]",arr[i][j]);
   }
   printf("\n");
 }
printf("the total no. of elements you have entered is:
                                %d\n",a*b);
```

```
for(i=0;i<a;i++)
 {
 for(j=0;j<b;j++)
  {
   if(arr[i][j]==0)
      zero++;
    else
      nzero++;
  }
}
int M=(a*b)/2;
if (zero>M)
 {
 printf("the matrix is a sparse matrix");
 printf("\n triplet: \n row\t column\t value\t");
 printf("\n. - - - - - - - - \n");
 for(int i=0;i<a;i++)
  {
   for(int j=0;j<b;j++)
    {
     if(arr[i][j]!=0)
       {
        printf("%d\t%d\t%d\n", i, j, arr[i][j]);
       }
    }
  }
}
else
  printf("the matrix is not a sparse matrix");
 return 0;
```

```
o/p -> Enter the no. of rows and columns of the matrix: 2
2
Enter 4 elements of the matrix: 1
2
3
4
the matrix entered is:
[1][2]
[3][4]
the matrix is not a sparse matrix.

Enter the no. of rows & columns of the matrix: 2
2
Enter 4 elements of the matrix: 1
0
0
0
the matrix entered is:
[1][0]
[0][0]
the total no. of elements in a sparse matrix entered is: 4
the matrix is a sparse matrix
triplet
row  column  value.
---  ------  -----
 0      0      1
```

EXPERIMENT-3

(1) WAP using C to create a stack & perform:
(i) push operation (ii) pop operation (iii) display operation.

```
#include<stdio.h>
int top = -1;
int push(int stack[], int n, int item)
{
  if(top == n-1)
   {
   printf("OVERFLOW");
   }
   else
    {
      top++;
      stack[top] = item;
    }
}
int pop(int stack[], int n)
{ int item;
   if(top == -1)
    {printf("UNDERFLOW");
     }
    else
    { item = stack[top];
      top--;
```

```
printf("popped item is %d", item);
 }
}
int display(int stack[])
{ int i;
 if(top==-1)
 { printf("stack is empty!");
 }
 else
{ printf("elements of stack are");
 for(i=top;i>0;i--)
   { printf("\n%d", stack[i]);
   }
}
}
int main()
 { int stack[20], n, item, option;
  printf("Enter the size of stack\n");
  scanf("%d", &n);
 do
  { printf("\n Enter your options");
  printf("\n 1.push\n2.pop\n3.Display\n4.exit
   scanf("%d", &option);
```

```
switch(option)
{
case 1:
    printf("Enter item to be pushed\n");
    scanf("%d", &item);
    push(stack, n, item);
    break;
case 2:
    pop(stack, n);
    break;
case 3:
    display(stack);
    break;
case 4:
    break;
default:
    printf("Invalid choice");

  }
} while(option!=4);
return 0;

}
```

O/P → Enter the size of stack:
4
Enter your option:
1. push
2. pop
3. Display
4. Exit
Enter item to be pushed
7
Enter your option.
1. push
2. pop
3. Display
4. Exit
1
Enter item to be pushed
8
Enter your option.
1. push
2. pop
3. Displayed
4. Exit.
3
Element of stack are
8
7

② write a c program to create a linear queue & perform the following options using UDF:
(i) Insertion (ii) Deletion and (iii) Traversal.

```
#include<stdio.h>
#include<stdlib.h>
#define size 1000
void enqueue();
void dequeue();
void show();
int inp_arr[size];
int rear=-1;
int front=-1;
int main()
{
  int ch;
  while(1)
  {
    printf("1. Enqueue operation\n");
    printf("2. Dequeue operation\n");
    printf("3. Display the queue\n");
    printf("4. Exit\n");
    printf("Enter your choice of operations: ");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1:
        enqueue();
        break;
```

```
case 2:
      dequeue();
      break;
case 3;
      show();
      break;
case 4;
      exit(0);
     default:
       printf("Incorrect choice\n");
    }
  }
}
void enqueue()
{
 int insert_item;
 if(rear==size-1)
   printf("Overflow\n");
 else
 {  if(front==-1)
      front=0;
  printf("Element to be inserted in the queue:");
  scanf("%d",&insert_item);
  rear=rear+1;
  inp_arr[rear]=insert_item;

 }
}
```

```
void dequeue()
{ if(front==-1 || front>rear)
   {
     printf("underflow\n");
     return 0;
   }
  else
  { printf("Element deleted from the queue : %d
           \n", inp_arr[front]);
    front=front+1;
  }
}
void show()
{
  if(front==-1)
    printf("empty queue\n");
  else
   {
     printf("queue:\n");
    for(int i=front; i<rear; i++)
       printf("%d", inp_arr[i]);
    printf("\n");
   }
}
```

O/P →
1. Enqueue operation.
2. Dequeue operation.
3. Display the queue.
4. Exit.

Enter your choice of operation : 1

Element to be inserted in the queue : 4

1. Enqueue operation
2. Dequeue operation
3. Display the queue
4. Exit

Enter your choice of operation: 1

Element to be inserted in the queue : 6

1. Enqueue operation.
2. Dequeue operation.
3. Display the queue
4. Exit.

Enter your choice of operation : 3

queue :

4 6

(3) write a C program to create a circular queue and perform the following operations using UDF: (i) insertion (ii) deletion and (iii) traversal.

```
#include<stdio.h>
#define SIZE 5
int items[SIZE];
int front=-1, rear=-1;
int isfull();
{
 if((front==rear+1)||(front==0&&rear=size-1))
    return 1;
  return 0;
}
int isempty()
{
  if(front==-1)
     return 1;
  return 0;
}
void enqueue(int element)
{
   if(isfull())
      printf("\n Queue is full!!\n");
    else
    {
      if(front==-1)
         front=0;
   rear=(rear+1)%size;
```

```
item[rear] = element;
printf("\n Inserted-> %d", element);
}
}
int dequeue()
{
  int element;
  if(isEmpty())
  {
   printf("\n Queue is empty !!\n");
   return(-1);
  }
  else
  {
   element = item[front];
   if(front == rear)
   {
     front = -1;
     rear = -1;
   }
   else
   {
    front = (front+1) % SIZE;
   }
   printf("\n Deleted element-> %d\n", element);
   return(element);
  }
}
```

```
void display()
{
 int i;
 if(isEmpty())
    printf("\n Empty Queue\n");
 else
   {
     printf("\n Front -> %d", front);
     printf("\n Items -> ");
     for(i=front; i!=rear; i=(i+1)%SIZE)
      {
       printf("%d", items[i]);
      }
     printf("%d", items[i]);
     printf("\n rear -> %d\n", rear());
   }
}
int main()
  {
    deQueue();
    enQueue(1);
    enQueue(2);
    enQueue(3);
    enQueue(4);
    enQueue(5);
    enQueue(6);
    display();
    deQueue();
    display();
    enQueue(7);
    display();
    enQueue(8);
    return 0;
  }
```

o/p → Queue is empty!!
Inserted → 1
Inserted → 2
Inserted → 3
Inserted → 4
Inserted → 5
Queue is full!!

Front → 0
Items → 1 2 3 4 5
Rear → 4
Deleted element → 1
Front → 1
Items → 2 3 4 5
Rear → 4
Inserted → 7
Front → 1
Items → 2 3 4 5 7
Rear → 0
Queue is full.

EXPERIMENT-4

① write a program to implement linear search on array elements using UDF.

```
#include<stdio.h>
int linearsearch(int arr, int n, int s_item)
{
 for(int i=0; i<=n; i++)
   { if(arr[i]==s_item)
      {
        return i;
      }
   }
   return -1;
}
int main()
{
 int n, s_item, totalnumber;
 int arr[50];
 printf("Enter the number of elements:");
 scanf("%d", &totalnumber);
 printf("enter the elements:");
 for(int i=0; i<totalnumber; i++)
  scanf("%d", &arr[i]);
 printf("the elements are as follow:");
 for(int i=0; i<totalnumber; i++)
   printf("arr[%d]: %d\n", i, arr[i]);
 printf("Enter the element u want to search");
```

```
scanf("%d", &s_item);
int p = linearsearch(a, totalnumber, s_item);
if (p != -1)
    printf("element %d found at index %d\n", s_item,
                                                p+1);
else
    printf("element %d not found in the array\n",
                                                s_item);
return 0;
}
```

O/P → Enter the no. of elements: 4
Enter the elements: 1
2
3
4
the elements are as follow:
arr[0] = 1
arr[1] = 2
arr[2] = 3
arr[3] = 4
enter the element u want to search: 1
element 1 found at index 1.

② write a program to implement binary search on array elements using UDF.

```
#include<stdio.h>
int bsearch(int list[], int lb, int ub, int item)
{ int mid, pos = -1;
  while(lb<=ub)
    {
      mid = (lb+ub)/2;
      if(list[mid]==item)
        return(mid+1);
      else if(list[mid]>item)
          ub= mid-1;
        else
          lb = mid+1;
    }
  return(pos);
}
int main()
{ int list[100], item, n, i;
  printf("enter the no. of elements:");
  scanf("%d",&n);
  printf("enter the elements:");
  for(i=0; i<n; i++)
    scanf("%d",&list[i]);
  printf("the items are as follows:");
  for(i=0; i<n; i++)
    printf("list[%d]:%d\n", i, list[i]);
```

```
printf("enter the item to search");
scanf("%d",&item);
int b = bsearch(list,0,n-1,item);
if (b==-1)
  printf("element is not present");
else
  printf("element is present at %d position",b);
return 0;
}
```

o/p →

enter the no. of element : 4
enter the element : 1
2
3
4
the item are as follows :
list[0] : 1
list[1] : 2
list[2] : 3
list[3] : 4
enter the item to search : 2
element 2 is present at 2 position.

LAB EXPERIMENT-5

write a c program to implement quicksort to a given to a given list of Integers to sort in ascending order.

```
#include<stdio.h>
void swap (int *a ,int* b)
{ int temp = *a;
   *a = *b;
   *b = temp;
}
int partition (int arr[], int low, int high)
{ int pivot = arr[high];
   int i = low-1;
for(int j= low; j<high; j++)
   {
   if (arr[j]<= pivot)
   { i++;
     swap (&arr[i], &arr[j]);
   }}
   swap (&arr[i+1], &arr[high]);
   return i+1;
}
void quicksort(int arr[], int low , int high){
   if(low<high){
      int pivotindex = partition(arr, low, high);
```

```
quicksort(arr, low, pivotIndex-1);
quicksort(arr, pivotIndex+1, high);
}
}
void printArray(int arr[], int size)
{ for(int i=0; i<size; i++)
{ printf("%d ", arr[i]);
}
printf("\n");
}
int main()
{ int arr[] = {9, 2, 5, 1, 7, 6, 8};
int size = sizeof(arr) / sizeof(arr[0]);
printf("original array: ");
printarray(arr, size);
quicksort(arr, 0, size-1);
printf("sorted array: ");
printarray(arr, size);
return 0;
}
```

o/p → original array: 9 2 5 1 7 6 8
sorted array: 1 2 5 6 7 8 9

LAB EXPERIMENT-6

write a 'C' program to create a structure called student to store rollno, name, age. create an array to input 5 students data and then create an UDF to display where age >= 20.

```
#include<stdio.h>
#define MAX_STUDENTS 5
// structure to student information.
struct student {
        int rollno;
        char name[50];
        int age;
       };
// function to display students with age >= 20.
void display students(struct student students[], int size)
 { printf("students with age >= 20:\n");
   printf("rollno\tname\tage\n");
  for(int i=0; i<size; i++)
    { if(students[i].age >= 20)
      { printf("%d\t%s\t%d\n", students[i].rollno,
                students[i].name, students[i].age);
      }
    }
 }
```

```
int main()
{
    struct student students[max_students];
    // input student data.
    printf("Enter student details:\n");
    for(int i=0; i<max_students; i++)
    {
        printf("student %d:\n", i+1);
        printf("roll no:");
        scanf("%d", &students[i].rollno);
        printf("name:");
        scanf("%s", students[i].name);
        printf("age:");
        scanf("%d", &students[i].age);
    }
// display students with age >= 20
display_students(students, max_students);
return 0;
}
```

o/p → Enter student details:

student 1:
rollno: 1
name: ram
age: 29

student 2:
roll no: 2
name: madhu
age: 20

student 3:
rollno: 3
name: deepak
age: 19

student 4:
rollno: 4
name: madhu
age: 18

student 5:
rollno: 5
name: deepak
age: 19

students with age >= 20

rollno	name	age
1	ram	29
2	madhu	20

LAB EXPERIMENT - 7

Write a C program to perform the operation on a single link list.

(i) Insertion at beginning (ii) Deletion at 1st node (iii) Display all nodes.

```
#include<stdio.h>
#include<stdlib.h>
// structure for a node in the linked list.
struct node{
            int data;
            struct node* next;
           };
// function to create a newnode.
struct node* create node(int data))
{
  struct node *newnode=(struct node*)malloc
                        (sizeof(struct node));
if(newnode==null)
{
 printf("memory allocation failed\n");
 exit(1);
}
newnode->data=data;
newnode->next=null;
return newnode;
}
//function to insert a node at the beginning of the
linked list.
void insertat beginning(struct node** head, int data)
{
```

NAME

```
struct node * newnode = create node(data);
if(*head == null)
{
 *head = new node;
}
else
{
newnode -> next = *head;
 *head = newnode;
}
printf("node with data %d inserted at the beginning
                                    \n", data);
}
//function to delet the first node in the linked list.
void delet first node (struct node **head)
{
if(*head == null)
{printf("linked list is already empty \n");}
else
{ struct node * temp = *head;
 *head = (*head) -> next;
 free(temp);
 printf("first node deleted: \n");
}
```

```
//function to display all the nodes in the linked list
void displaylist(struct node* head)
{
  if(head==null)
   {
    printf("linked list is empty\n");
   }
 else
   { printf("linked list nodes:");
     while(head!=null)
{
  printf("%d", head->data);
   head=head->next.
  }
  printf("\n");
}
}
int main()
 { struct node *head=null;
insert A - beginning across(&head,3);
insert B - at beginning(&head,*);
insert at beginning(&head,1);
struct node* intersectionnode = create node(5);
intersection node->next=next.
head=intersectionnode.
printf("intersection at the beginning\n");
return 0;
}
```

0/p → node with data 3 inserted at the beginning.
node with data 1 inserted at the beginning.
node with 1 inserted at the beginning.

linklist nodes: 1 1 3
Intersection at the beginning.
linklist nodes: 5 1 1 3
first node deleted.
linklist nodes: 1 1 3