DATA STRUCTUTES LAB EXAM

SUBMITTED BY:

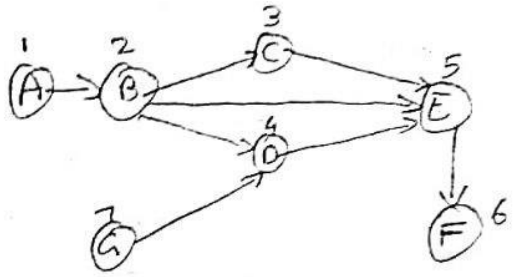
ASWIN PRABHAKARAN

S1 MCA

ROLL NO:13

Qno 1:

(a) Consider a directed acyclic graph G given in the following figure



Develop a program to implement topological sorting

$$\begin{array}{c|ccccccc}
 & A & B & C & D & E & F & G \\
\hline
A & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
C & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
D & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
E & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}$$

$\Rightarrow \{ 1\ 7\ 2\ 3\ 4\ 5\ 6 \}$

Algorithm for Topological sort

```
1. Begin .
  2. mark v as visited
  3. for all vertices v which is adjacent with v, do
  4. if v is not visited , then
          topsort ( c, visited, stack)
  5. done
  6. push v into a stack
7. End
```

operation topsort (Graph)

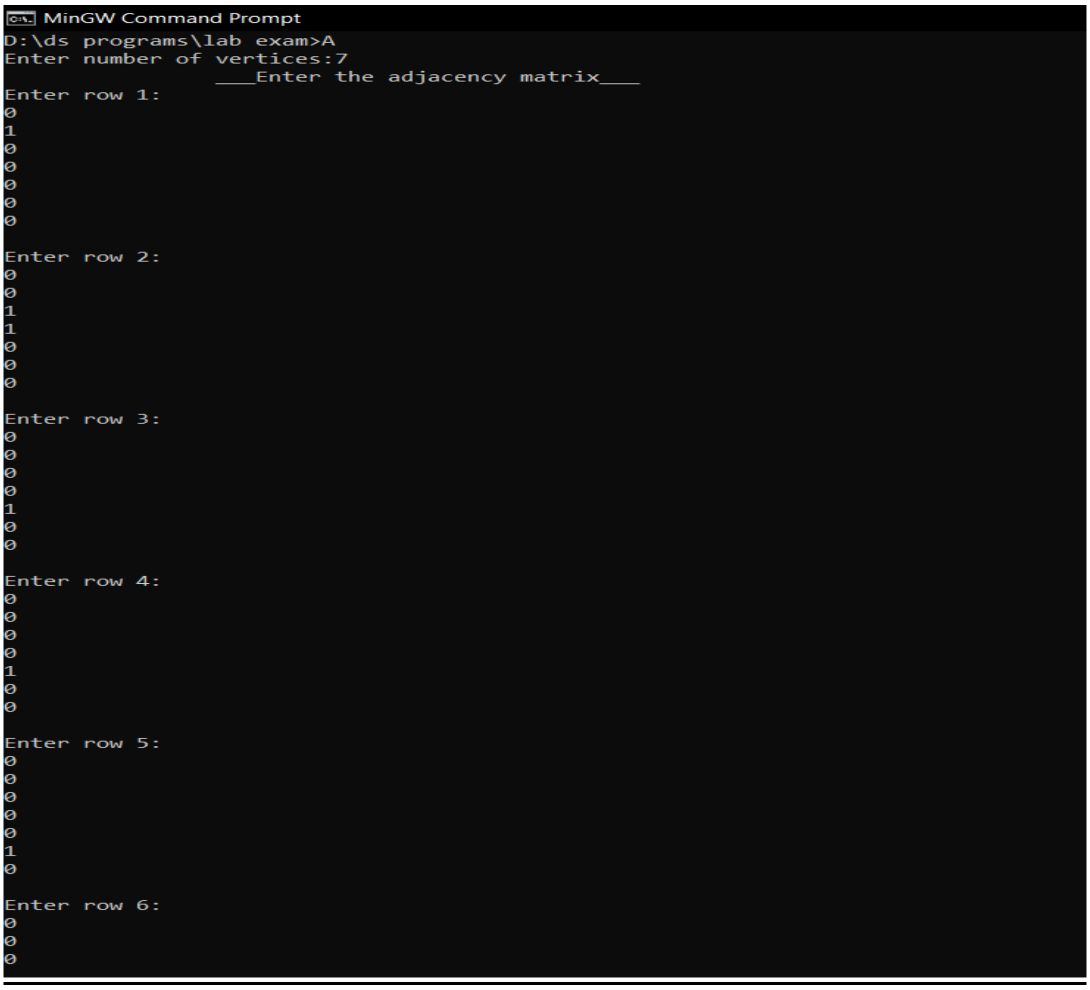
```
Begin
    Intially mark all nodes as unvisited
    for all nodes v of the graph, do
        if v is not visited , then
            topsort ( i , visited , stack)
  done
  pop and print all the elements in the
    stack .
End .
```

CODE

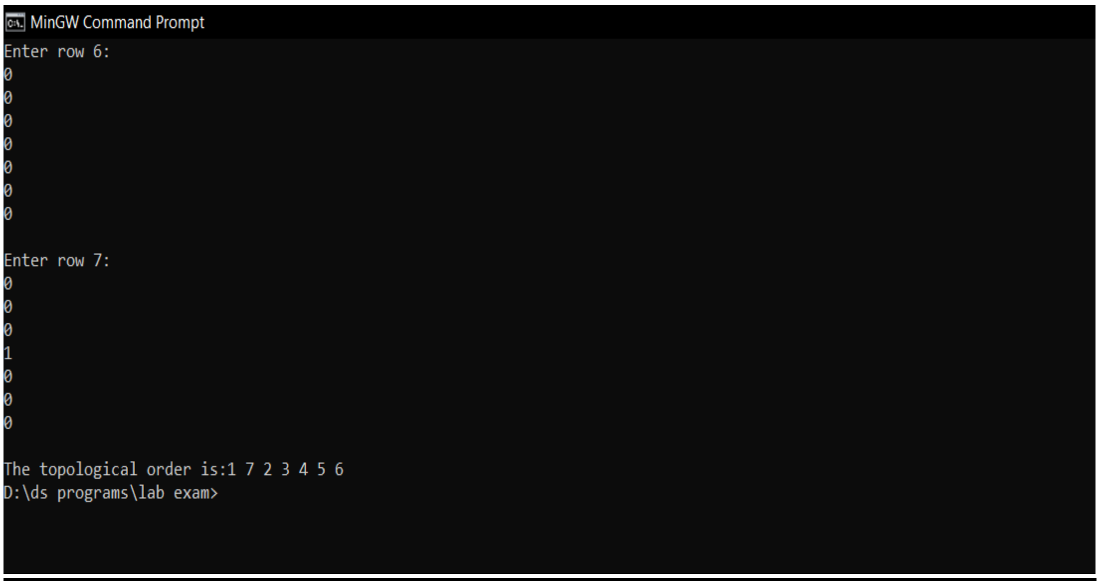
```
#include<stdio.h>
int main(){
int i,j,k,n,a[10][10],index[10],flag[10],count=0;
printf("Enter number of vertices:");
scanf("%d",&n);
printf("\t\t___Enter the adjacency matrix___");
for(i=0;i<n;i++){
              printf("\nEnter row %d:\n",i+1);
              for(j=0;j<n;j++)
                     scanf("%d",&a[i][j]);
       }

       for(i=0;i<n;i++){
    index[i]=0;
    flag[i]=0;
  }
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      index[i]=index[i]+a[j][i];

  printf("\nThe topological order is:");

  while(count<n){
    for(k=0;k<n;k++){
      if((index[k]==0) && (flag[k]==0))
       {
        printf("%d ",(k+1));
```

```
                flag [k]=1;
            }

            for(i=0;i<n;i++){
                if(a[i][k]==1)
                    index[k]--;
            }
        }

        count++;
    }

    return 0;
}
```

OUTPUT



```
MinGW Command Prompt
D:\ds programs\lab exam>A
Enter number of vertices:7
                ___Enter the adjacency matrix___
Enter row 1:
0
1
0
0
0
0
0

Enter row 2:
0
0
1
1
0
0
0

Enter row 3:
0
0
0
0
1
0
0

Enter row 4:
0
0
0
0
1
0
0

Enter row 5:
0
0
0
0
0
1
0

Enter row 6:
0
0
0
```



```
MinGW Command Prompt
Enter row 6:
0
0
0
0
0
0
0

Enter row 7:
0
0
0
1
0
0
0

The topological order is:1 7 2 3 4 5 6
D:\ds programs\lab exam>
```

Qno 2:

(a) Write a Program for creating DLL and perform the following operations

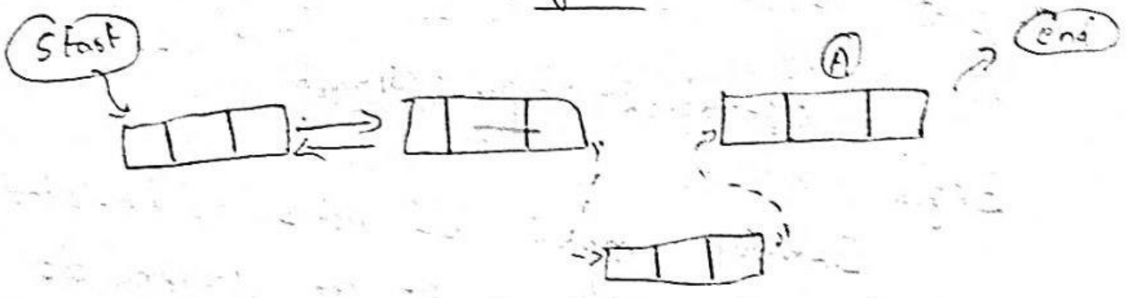
(a) Insert an element at a particular position

(b) Search an element

(c) Delete an element at the end of the list.

Algorithm

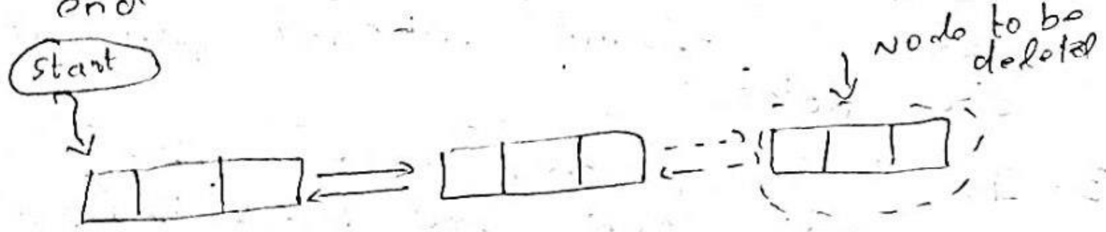
① Inserting an element at at user wanting place



```
step1:  Begin
step2:  create a node (NEW NODE)
            NEWNODE → DATA = VALUE
            NEWNODE → PREVIOUS = NULL
            NEWNODE → NEXT = NULL
step3:      PTR = START
            WHILE (PTR → NEXT != CHOICE)
                {  PTR = PTR → NEXT
                }
```

```
step 4: NEWNODE → NEXT = A { PTR → NEXT }
step 5: NEWNODE → PREV = PTR.
step 6: A → PREV = NEWNODE.
step 7: PTR → NEXT = NEWNODE.
step 8: Exit.
```

② Deletion of an element at the end of the list



```
step 1: Begin
step 2: PTR = START.
step 3: WHILE (PTR → NEXT != NULL)
        {
            PTR = PTR → NEXT
        }
step 4: TEMP = PTR → PREV
step 5: TEMP → NEXT = NULL
step 6: FREE (PTR)
step 7: Exit
```

③ searching an element

Algorithm.

1. Declare a temp pointer and initialize it to the head of the list

2. Iterate the loop until temp reaches start address (last node in the list, as in a circular fashion). check for n element whether present or not.

3. If it is present, raise a flag, increment count and break the loop.

4. At last, the last node is not visited yet check for the n element if present repeat step 3.

CODE

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

   struct node *prev;

   struct node *next;

   int data;

};

struct node *head;

void create();

void insert_spec();

void delet_last();

void display();

void search();

void main()

{

   int choice = 0;

   while (choice != 9)

   {

      printf("\n*********Main Menu*********");

      printf("\nChoose an option from the following list");

      printf("\n1.Create a linked list\n2.Insert at any random location\n3.Delete from last
pos\n4.Searching\n5.Display\n6.Exit\n");

      printf("\n Please enter your choice? = ");

      scanf("%d", &choice);

      switch (choice)

      {

      case 1:

         create();

         break;
```

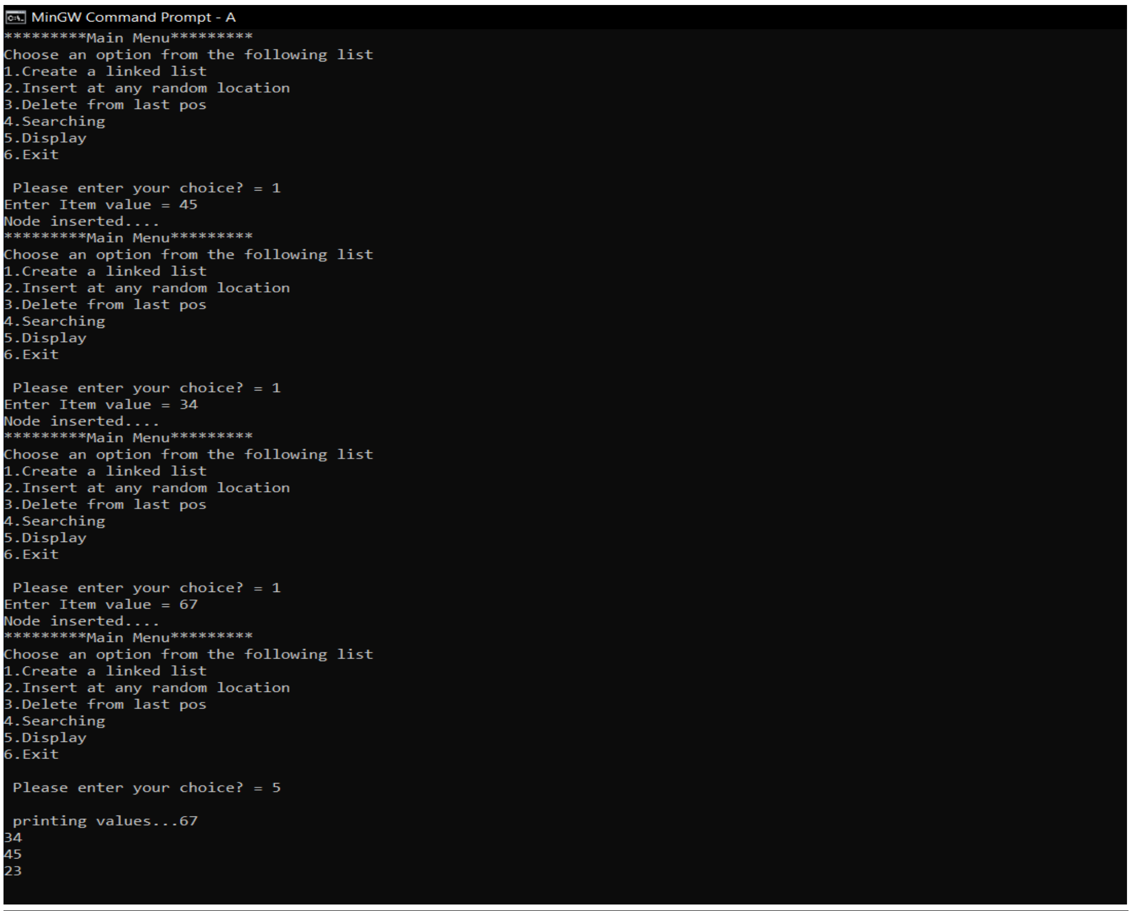
```
        case 2:
            insert_spec();
            break;
        case 3:
            delet_last();
            break;
        case 4:
            search();
            break;
        case 5:
            display();
            break;
        case 6:
            exit(0);
            break;
        default:
            printf("Please enter  a valid choice.......");
        }
    }
}
void create()
{
    struct node *ptr;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("Enter Item value = ");
        scanf("%d", &item);

        if (head == NULL)
        {
            ptr->next = NULL;
            ptr->prev = NULL;
            ptr->data = item;
            head = ptr;
        }
        else
        {
            ptr->data = item;
            ptr->prev = NULL;
            ptr->next = head;
            head->prev = ptr;
            head = ptr;
        }
        printf("Node inserted....");
    }
}

void insert_spec()
{
    struct node *ptr, *temp;
    int item, loc, i;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL)
    {
        printf("\n OVERFLOW");
    }
```

```
   else
   {
      temp = head;
      printf("Enter the location = ");
      scanf("%d", &loc);
      for (i = 0; i < loc; i++)
      {
         temp = temp->next;
         if (temp == NULL)
         {
            printf("\n There are less than %d elements",&loc);
            return;
         }
      }
      printf("Enter value = ");
      scanf("%d", &item);
      ptr->data = item;
      ptr->next = temp->next;
      ptr->prev = temp;
      temp->next = ptr;
      temp->next->prev = ptr;
      printf("\n Node inserted at the current location....);
   }
}

void delet_last()
{
   struct node *ptr;
   if (head == NULL)
   {
      printf("\n UNDERFLOW");
```

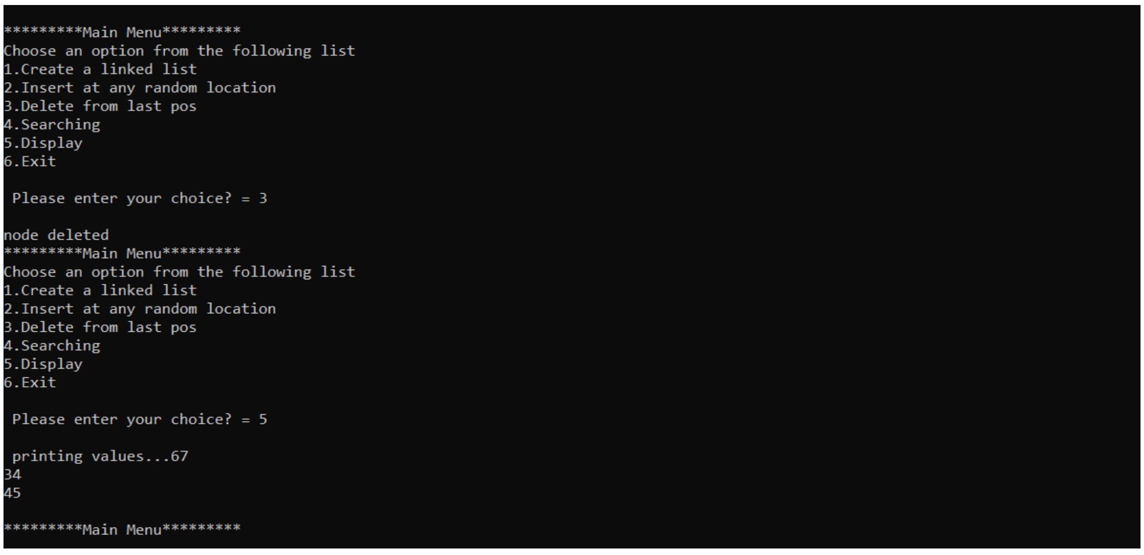
```
    }
    else if (head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\n Node Succesfully deleted........");
    }
    else
    {
        ptr = head;
        while (ptr->next != NULL)
        {
            ptr = ptr->next;
        }
        ptr->prev->next = NULL;
        free(ptr);
        printf("\nnode deleted");
    }
}
void display()
{
    struct node *ptr;
    printf("\n printing values...\n");
    ptr = head;
    while (ptr != NULL)
    {
        printf("%d\n", ptr->data);
        ptr = ptr->next;
    }
}
void search()
```

```
{
   struct node *ptr;
   int item, i = 0, flag;
   ptr = head;
   if (ptr == NULL)
   {
      printf("\nEmpty List");
   }
   else
   {
      printf("\nEnter an item  you want to search?");
      scanf("%d", &item);
      while (ptr != NULL)
      {
         if (ptr->data == item)
         {
            printf("\nItem found at location %d ", i + 1);
            flag = 0;
            break;
         }
         else
         {
            flag = 1;
         }
         i++;
         ptr = ptr->next;
      }
      if (flag == 1)
      {
         printf("\nItem not found");
      }}}
```

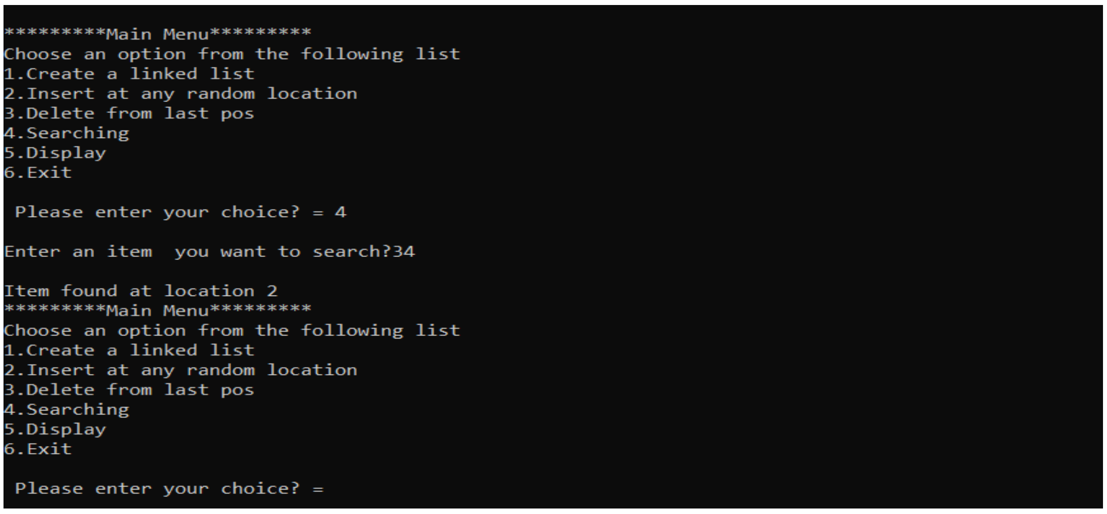
OUTPUT


MinGW Command Prompt - A

```
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? = 1
Enter Item value = 45
Node inserted....
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? = 1
Enter Item value = 34
Node inserted....
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? = 1
Enter Item value = 67
Node inserted....
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? = 5

 printing values...67
34
45
23
```



```
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? = 3

node deleted
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? = 5

 printing values...67
34
45

*********Main Menu*********
```


```
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? = 4

Enter an item  you want to search?34

Item found at location 2
*********Main Menu*********
Choose an option from the following list
1.Create a linked list
2.Insert at any random location
3.Delete from last pos
4.Searching
5.Display
6.Exit

 Please enter your choice? =
```