Rajshahi University of Engineering & Technology
Department of Computer Science of Engineering

EXPERIMENT NO: 04
NAME OF EXPERIMENT: Linked Lists

SUBMITTED TO:

RIZOAN TOUFIQ
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

SUBMITTED BY:

NAME: MD. ARIFUL ISLAM
ROLL NO.: 1803046
GROUP: 2ND THIRTY
DATE OF EXP.: 23-09-2019
DATE OF SUB. : 30-09-2019
SERIES: 18 SERIES

MACHINE CONFIGURATION:

ASUS X510UF
CORE I5 8TH GEN PROCESSOR
UP TO 3.4 GHZ
8 GB RAM
OS WIN 10

INDEX

THEORY: DATA structures are classified as either linear or nonlinear. A data structure is said to be linear if its elements form a sequence or in other words , a linear list. There are two basic ways of representing such linear structures in memory :

1. **Array**.
2. **Linked list**.

A linked list or one-way list is a linear collection of data elements,called **nodes**, where the linear order is given by means of pointers. Each node is divided into two parts :

1. The first part contains the **information** of the elements.
2. The second part called the **link field** or **nextpointer field**,contains the address of the next node in the list.

The operations normally performed on any linear structures are :

1. **Traversal :** Processing each elements in the list.

2. **Search :** Finding the location of an element.

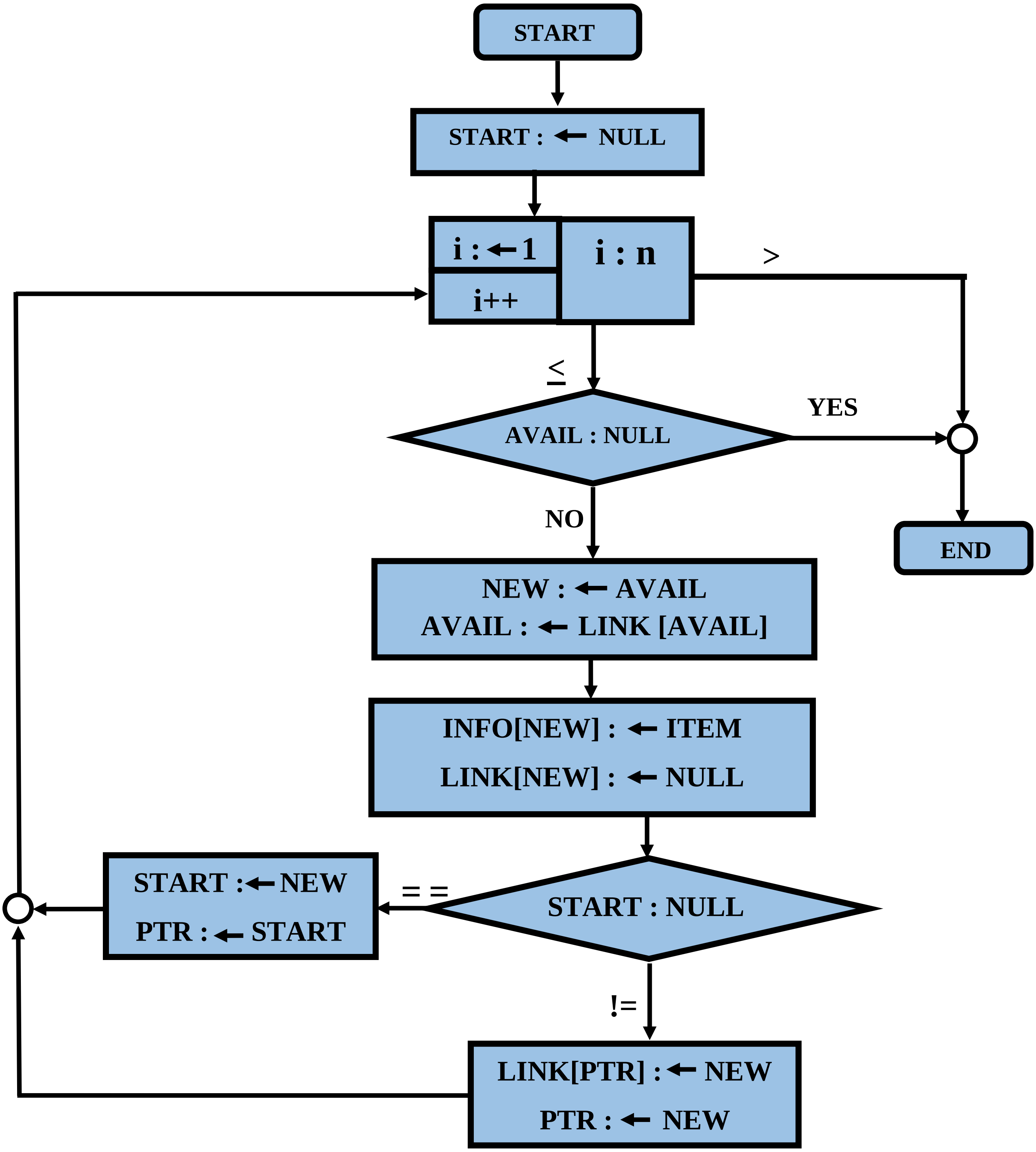
3. **Insertion :** Adding a new element to the list.

4. **Deletion :** Removing an element from the list.

Etc.

PROBLEM 1: Creating a linked list.

FLOW CHART:



ALGORITHM:

(Creating a Linked List) This algorithm create a linked list with n nodes.

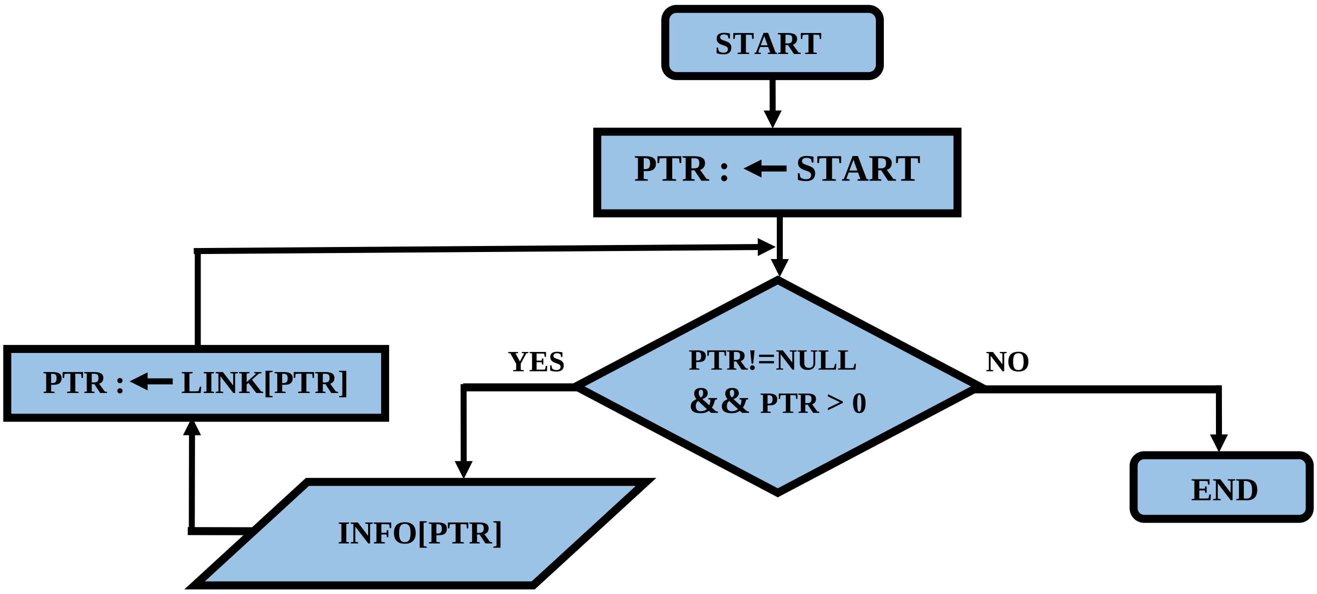
```
1. ST := NULL
2. Repeat Steps 3 to 5 for I = 1 to N
           [OVERFLOW] If AVAIL = NULL, then:
                 Write: OVERFLOW, and Exit.
3. [Remove first node from AVAIL.]
     Set NEW:=AVAIL and AVAIL := LINK[AVAIL]
4.  Set INFO[NEW] := ITEM and LINK[NEW] := NULL
5. If START = NULL, then: Set ST := NEW and PTR : = ST
       Else:
            Set LINK[PTR] := NEW and PTR = NEW
       [End of If structure]
6.  Exit.
```

CODE:

```
#include<stdio.h>

char info[15];
int link[10]={2,3,4,5,6,7,8,9,1,-1};
int start=-1;
int avail=0;

int newnode(){
        int newindex;
        if(avail==-1){
        printf("\nOverflow\n");
        return -1; }
        else{
        newindex=avail;
        avail=link[avail];
        return newindex;  }
}
void creat_list(){
        int ptr=-1,i,newindex;
        char ch='H';
        for(i=0;i<10;i++){
        newindex=newnode();
        if(newindex==-1)
        break;
        info[newindex]=ch;
        link[newindex]=-1;
        if(start==-1){
        start=newindex;
        ptr=newindex;  }
        else{
        link[ptr]=newindex;
        ptr=newindex;  }
        ch++;
}

int main(){
        creat_list();
        return 0;
}
```

PROBLEM 2: Traversing a linked list.

FLOW CHART:



ALGORITHM:

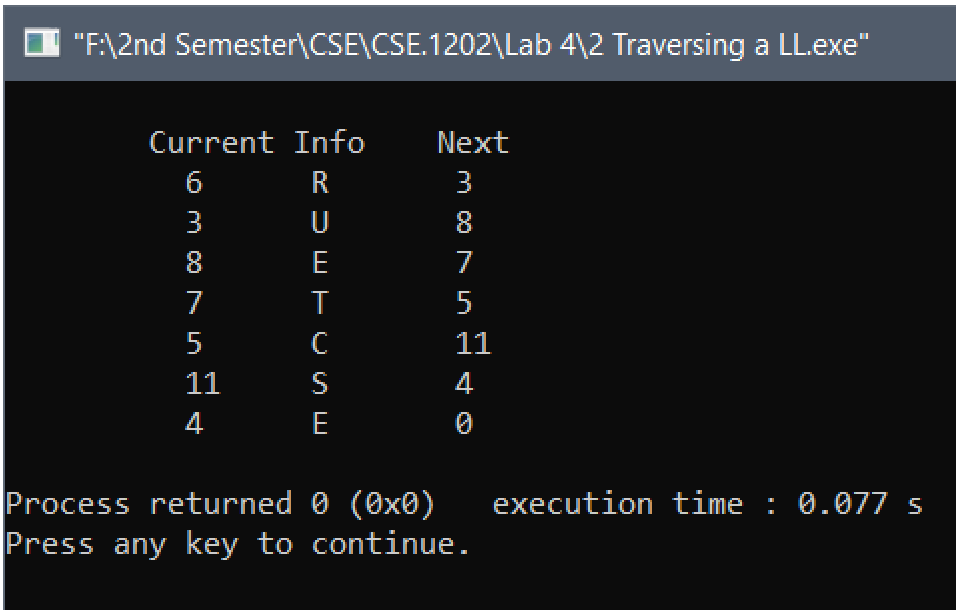
LIST is a linked list in memory. This algorithm traverses LIST, applying an operation PROCESS to each element of LIST. The variable PTR points to a node currently being processed.

```
1. Set PTR:=START
2. Repeat steps 3 and 4 while PTR≠NULL
3.        Apply PROCESS to INFO[PTR]
4.        SET PTR := LINK[PTR]
       [End of Repeat 2 loop]
5. Exit.
```

CODE:

```
#include<stdio.h>

int main()
{
    char info[12]={'\0','\0','U','E','C','R','T','E','\0','\0','S','\0'};
    int link[12]={'\0','\0',8,'\0',11,3,5,7,'\0','\0',4,'\0'};
    int start=6,ptr;

    ptr=start-1;
    printf("\n\tCurrent\tInfo\tNext\n");
    while(ptr!='\0'&&ptr>0)
    {
        printf("\t  %d \t %c \t %d\n",ptr+1,info[ptr],link[ptr]);
        ptr=link[ptr]-1;
    }

    return 0;
}
```

Output :

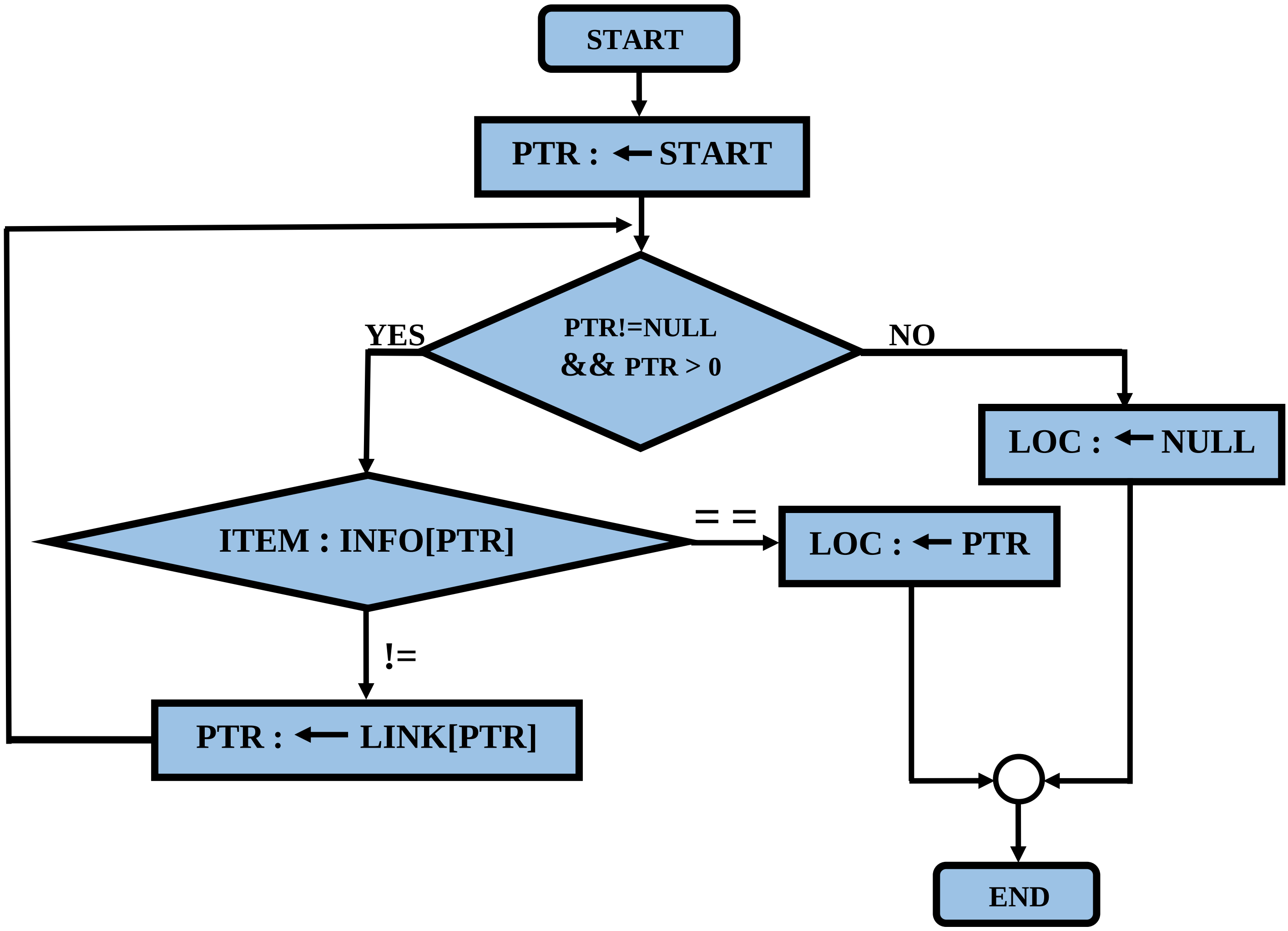


"F:\2nd Semester\CSE\CSE.1202\Lab 4\2 Traversing a LL.exe"

```

        Current Info    Next
          6      R        3
          3      U        8
          8      E        7
          7      T        5
          5      C        11
          11     S        4
          4      E        0

Process returned 0 (0x0)   execution time : 0.077 s
Press any key to continue.
```

PROBLEM 3: Searching a linked list (LIST is unsorted).

FLOW CHART:



ALGORITHM: SEARCH (INFO, LINK, START, ITEM, LOC)

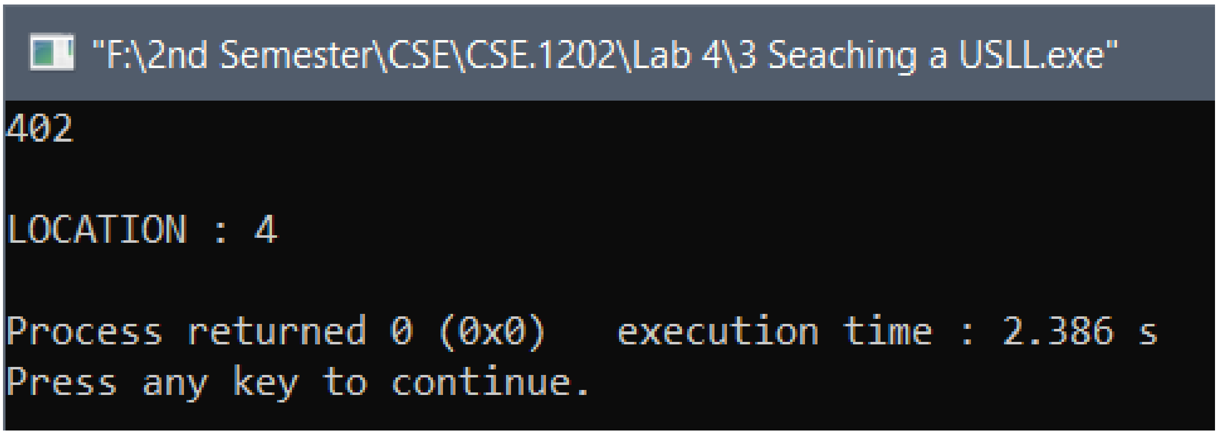
LIST is a linked list in memory. This algorithm finds the location LOC of the node where ITEM first appears in LIST, or sets LOC-NULL.

```
1. Set PTR:= START
2. Repeat steps 3 and 4 while PTR≠NULL
3. If ITEM = INFO[PTR] then:
           Set LOC:=PTR and Exit.
       Else:
           SET PTR := LINK[PTR]
       [End of If statement]
   [End of Repeat 2 loop]
4. [Search is unsuccessful] Set LOC:=NULL
5. Exit.
```

CODE:

```
#include<stdio.h>

int main(){
    int info[12]={'\0','\0',201,402,325,101,301,251,'\0','\0',385,'\0'};
    int link[12]={'\0','\0',8,'\0',11,3,5,7,'\0','\0',4,'\0'};
    int start=6,ptr,item,loc=-1;

    scanf("%d",&item);
    ptr=start-1;
    while(ptr!='\0'&&ptr>0){
        if(item==info[ptr]){
            loc=ptr+1;
            break;
        }
        else
            ptr=link[ptr]-1;
    }
    printf("\n%d\n",loc);

    return 0;
}
```

OUTPUT :

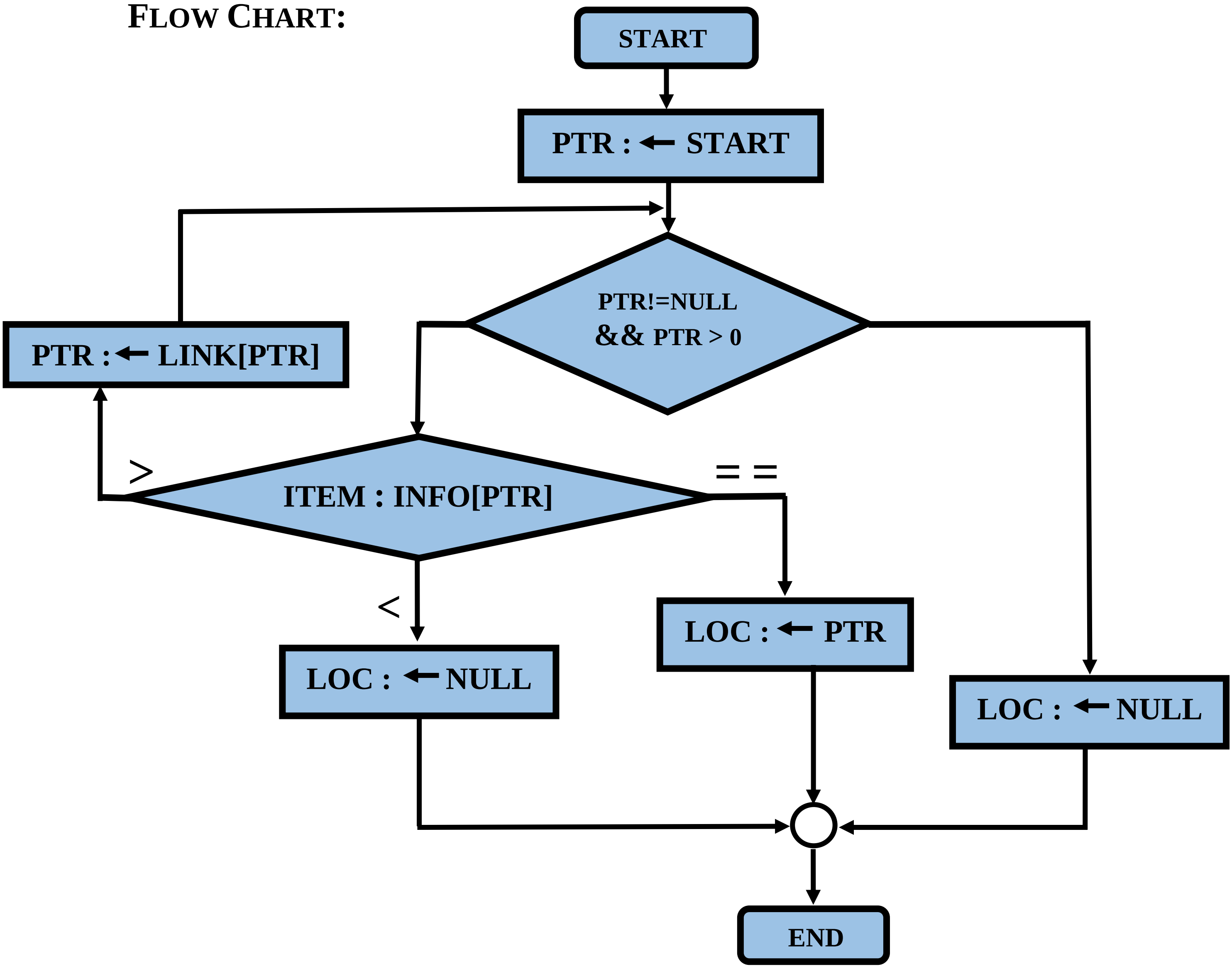


"F:\2nd Semester\CSE\CSE.1202\Lab 4\3 Seaching a USLL.exe"

```
402

LOCATION : 4

Process returned 0 (0x0)   execution time : 2.386 s
Press any key to continue.
```

PROBLEM 4: Searching a linked list (Sorted in ascending order).

FLOW CHART:



ALGORITHM: SRCHSL (INFO, LINK, START, ITEM, LOC)

LIST is a sorted linked list in memory. This algorithm finds the location LOC of the node where ITEM first appears in LIST, or sets LOC-NULL.

```
1. Set PTR:= START
2. Repeat steps 3 and 4 while PTR≠NULL
3. If ITEM<INFO [PTR], then: Set PTR:= LINK[PTR]
          Else if ITEM = INFO [PTR] then:
                Set LOC:=PTR and Exit.
```

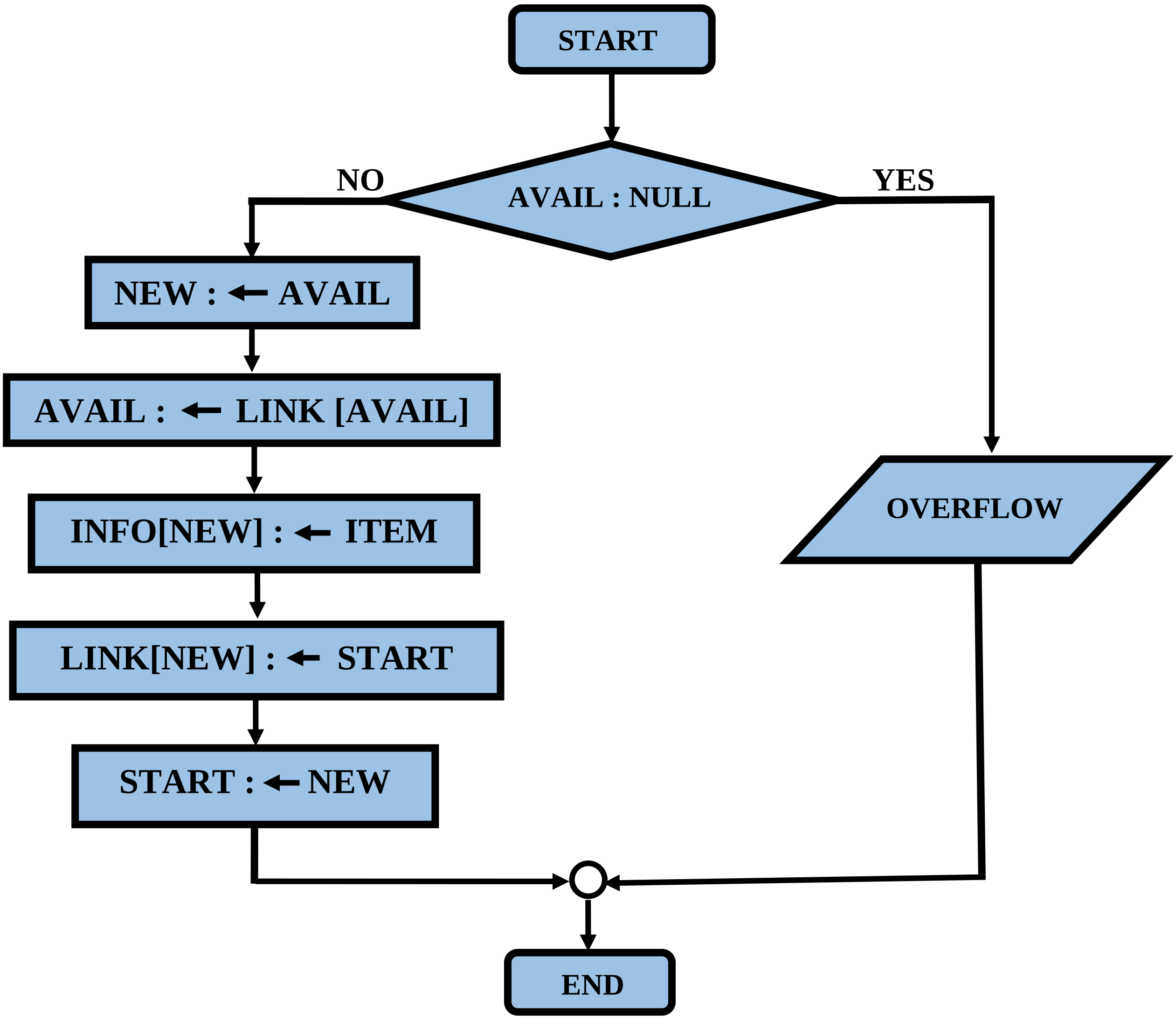
```
                    Else:
                         SET LOC:= NULL, and Exit.
                              [End of If statement]
                    [End of Repeat 2 loop]
          4. [Search is unsuccessful] Set LOC:=NULL
          5. Exit.
```

CODE:

```
#include<stdio.h>

int main(){
    int info[12]={'\0','\0',201,402,325,101,301,251,'\0','\0',385,'\0'};
    int link[12]={'\0','\0',8,'\0',11,3,5,7,'\0','\0',4,'\0'};
    int start=6,ptr,item,loc=-1;

    scanf("%d",&item);
    ptr=start-1;
    while(ptr!='\0'&&ptr>0){
        if(item>info[ptr])
            ptr=link[ptr]-1;
        else if(item==info[ptr]){
            loc=ptr+1;
            break;
        }
        else{
            loc=-1;
            break;
        }
    }
    if(loc==-1)
        printf("\nData not Found\n");
    else
        printf("\nLOCATION: %d\n",loc);

    return 0;
}
```

PROBLEM 5: Insertion into a Linked List (beginning of a list).

FLOW CHART:



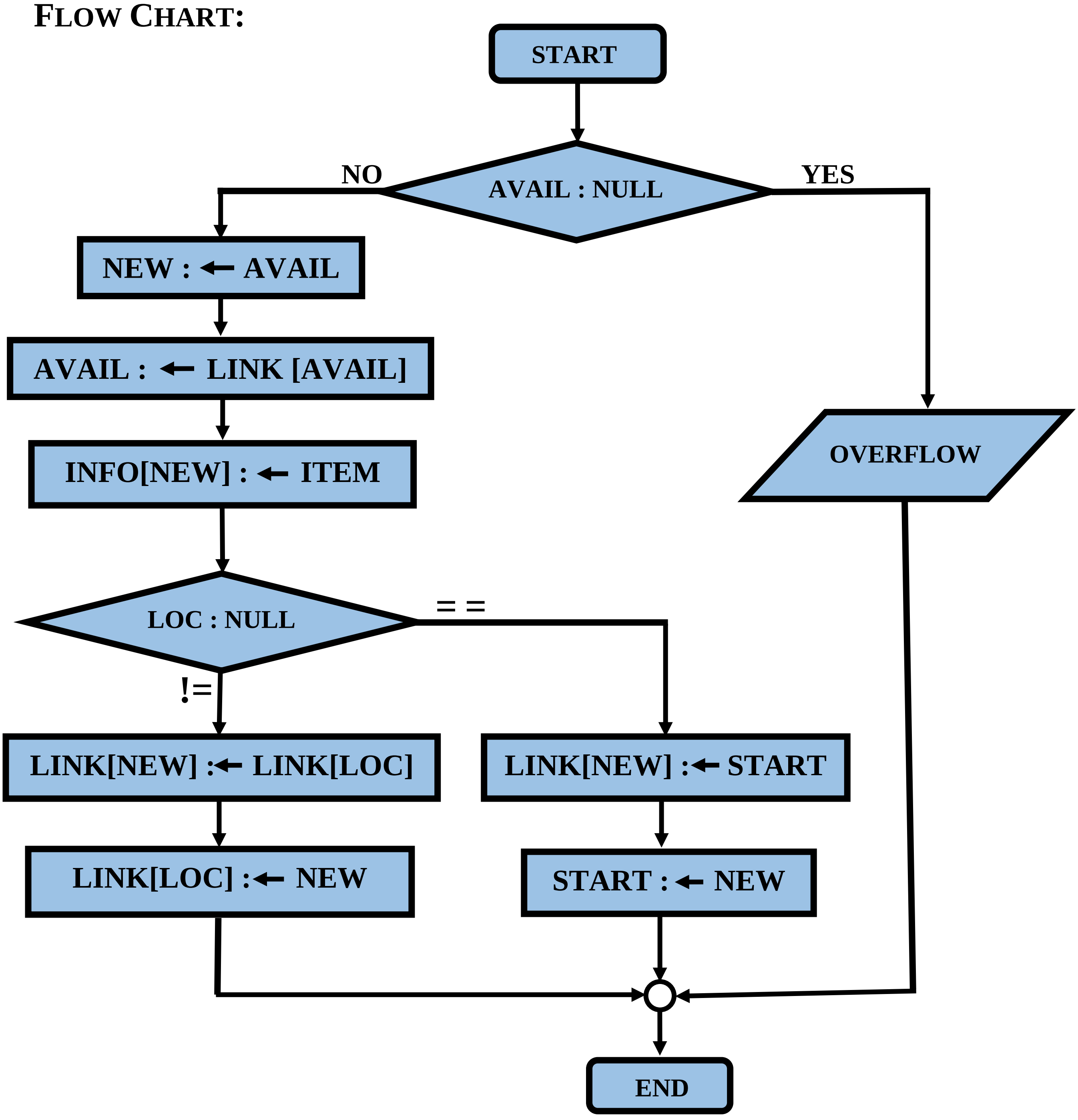
ALGORITHM: INSFIRST (INFO, LINK, START, AVAIL, ITEM)

This algorithm inserts ITEM as the first node in the list.

```
1. [OVERFLOW?] If AVAIL = NULL, then Write: OVERFLOW, and Exit.
2. [Remove first node from AVAIL list]
        Set NEW := AVAIL and AVAIL:= LINK [AVAIL]
3. Set INFO [NEW] := ITEM.
4. Set LINK [NEW] := START.
5. Set START := NEW.
6. Exit.
```

PROBLEM 6: Insertion into a Linked List (After a given node).

FLOW CHART:



ALGORITHM: INSLOC (INFO, LINK, START, AVAIL, LOC, ITEM)

This algorithm inserts ITEM so that ITEM follows the node with location LOC or insert ITEM as the first node when LOC = NULL.

1. [OVERFLOW?] If AVAIL = NULL, then Write: OVERFLOW, and Exit

2. [Remove first node from AVAIL list]

```
        Set NEW := AVAIL and AVAIL:= LINK [AVAIL]
3. Set INFO [NEW] := ITEM.
4. If LOC = NULL, then:
                Set LINK [NEW] := START and START := NEW.
        Else: Set LINK [NEW] := LINK [LOC] and LINK [LOC] := NEW.
   [End of If statement]
5. Exit
```

DISCUSSION: The problems were based on Linked List. After trying sometime I had solved the first four problems. But still **I have some confussion** in problem no. **5 , 6 & 7** and for this I could not complete these problems.

The End