**ASSIGNMENT – 13**

**1.PROBLEM STATEMENT**

Write a program in C to merge two sorted lists into a single list.

**2.ALGORITHMS**

Algorithm **Create\_Node**

**Input:** The data ‘item’ which is to be inserted into the node

**Output:** A node constructed in memory with item inserted as the data

**Remarks:** Node is allocated only if sufficient memory is available

**Steps:**

1. A node is allocated in memory and it’s address is stored in a pointer ptr
2. ptr.DATA=item
3. ptr.LINK=null
4. Return **ptr**
5. **Stop**

Algorithm **Insert\_Node**

**Input:** The address ‘start’ of the first node of the list and the data ‘item’ to be inserted

**Output:** A node inserted into the ‘start’ linked list with ‘item’ stored in it.

**Remarks:** Node is allocated only if sufficient memory is available.

**Steps:**

1. Ptr=**Create\_Node**(item)
2. **If**(start=null) **then** //for an empty list
3. start=ptr //ptr becomes start
4. end=ptr //ptr becomes end
5. **Else**
6. end.LINK=ptr //point end link to ptr
7. end=ptr //update end to ptr
8. **EndIf**
9. **Return** start

Algorithm **Merge\_Lists**

**Input:** The address of the first nodes both the lists, namely ‘start1’ and ‘start3’ and the address of the first node ‘start3’ of the linked list in which the merged result is to be stored.

**Output:** The contents of the the list ‘start1’ and ‘start2’ merged in ascending order within ‘start3’.

**Remarks:** The lists ‘start1’ and ‘start2’ must be in sorted order.

**Steps:**

1. Temp1=start1,temp2=start2
2. **While**(temp≠null **AND** temp≠null) **do** //untill either list exhausts
3. **If**(temp1.DATA≤temp2.DATA) **then** //temp1.DATA is greater or equal
4. start3=**Insert\_Node**(start3,temp1.DATA) //insert in start3
5. Temp1=temp1.LINK //shift temp1 to next node
6. **Else**
7. Start3=**Insert\_Node**(start3,temp2.DATA)
8. Temp2=temp2.LINK
9. **EndIf**
10. **EndWhile**
11. **If**(Temp1=null) **then** //when first list exhausts
12. **While**(temp2≠null) **do** //copy the second list as it is
13. Start3=**Insert\_Node**(start3,temp2.DATA)
14. **EndWhile**
15. **Else**
16. **While**(temp1≠null) **do** //copy the first list as it is
17. Start3=**Insert\_Node**(start3,temp1.DATA)
18. Temp1=temp1.LINK
19. **Endwhile**
20. **EndIf**
21. **Return** start3
22. Stop

**3.SOURCE CODE**

#include<stdlib.h>

#include<stdio.h>

//structure of a node of a linked list

typedef struct node

{

    int data;

    struct node \*link;

}node;

//function to create a new node in memory

node\* createnode(int item)

{

    node\* ptr;

    ptr=(node\*)malloc(sizeof(node));

    ptr->data=item; //store item in data part

    ptr->link=NULL; //initialise ptr link to NULL

    return ptr;

}

//function to display a linked list

void displist(node \*start)

{

    node \*temp;

    temp=start;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->link;

    }

}

node \*end; //global pointer to point to the last element of the list

//function to insert a new node at the end of a linked list

node\* insertnode(node\* start,int item)

{

    node \*ptr,\*temp=start;

    ptr=createnode(item);

    if(start==NULL) //if list is empty

{

        start=ptr;

end=ptr

}

    else

    {

        end->link=ptr //insert ptr at end

end=ptr //update end

    }

    return start;

}

//function to take input in a list

node\* getlist(node \*start,int len)

{

    int i,item;

    printf("Enter %d elements: ",len);

    for(i=0;i<len;i++)

    {

        scanf("%d",&item);

        start=insertnode(start,item);

    }

    return start;

}

//function to merge two sorted lists into a single list

node\* mergelist(node\* start1,node\* start2,node\* start3)

{

    node \*temp1=start1,\*temp2=start2;

    int i,j,k;

    while(temp1!=NULL && temp2!=NULL) //while both lists don’t exhaust

        {

            if((temp1->data)<=(temp2->data))

            {

                start3=insertnode(start3,temp1->data);

                temp1=temp1->link;

            }

            else

            {

                start3=insertnode(start3,temp2->data);

                temp2=temp2->link;

            }

        }

    if(temp1==NULL) //if list1 ends while list2 still remains

    {

        while(temp2!=NULL) // copy all the data of list2 to merged list

        {

            start3=insertnode(start3,temp2->data);

            temp2=temp2->link;

        }

    }

    else // if list2 ends while list1 still remains

    {

        while(temp1!=NULL) //copy all data of list1 to merged list

        {

            start3=insertnode(start3,temp1->data);

            temp1=temp1->link;

        }

    }

    return start3;

}

int main(void)

{

    node \*start1=NULL,\*start2=NULL,\*start3=NULL;

    int len1,len2,len3=len1+len2;

printf("To merge two sorted lists into a single list: \n");

    printf("Enter the length of the first list: ");

    scanf("%d",&len1);

    start1=getlist(start1,len1);

    printf("\nEntered List: ");

    displist(start1);

    printf("\nEnter the length of the second list: ");

    scanf("%d",&len2);

    start2=getlist(start2,len2);

    printf("\nEntered List: ");

    displist(start2);

    start3=mergelist(start1,start2,start3);

    printf("\nMerged List: ");

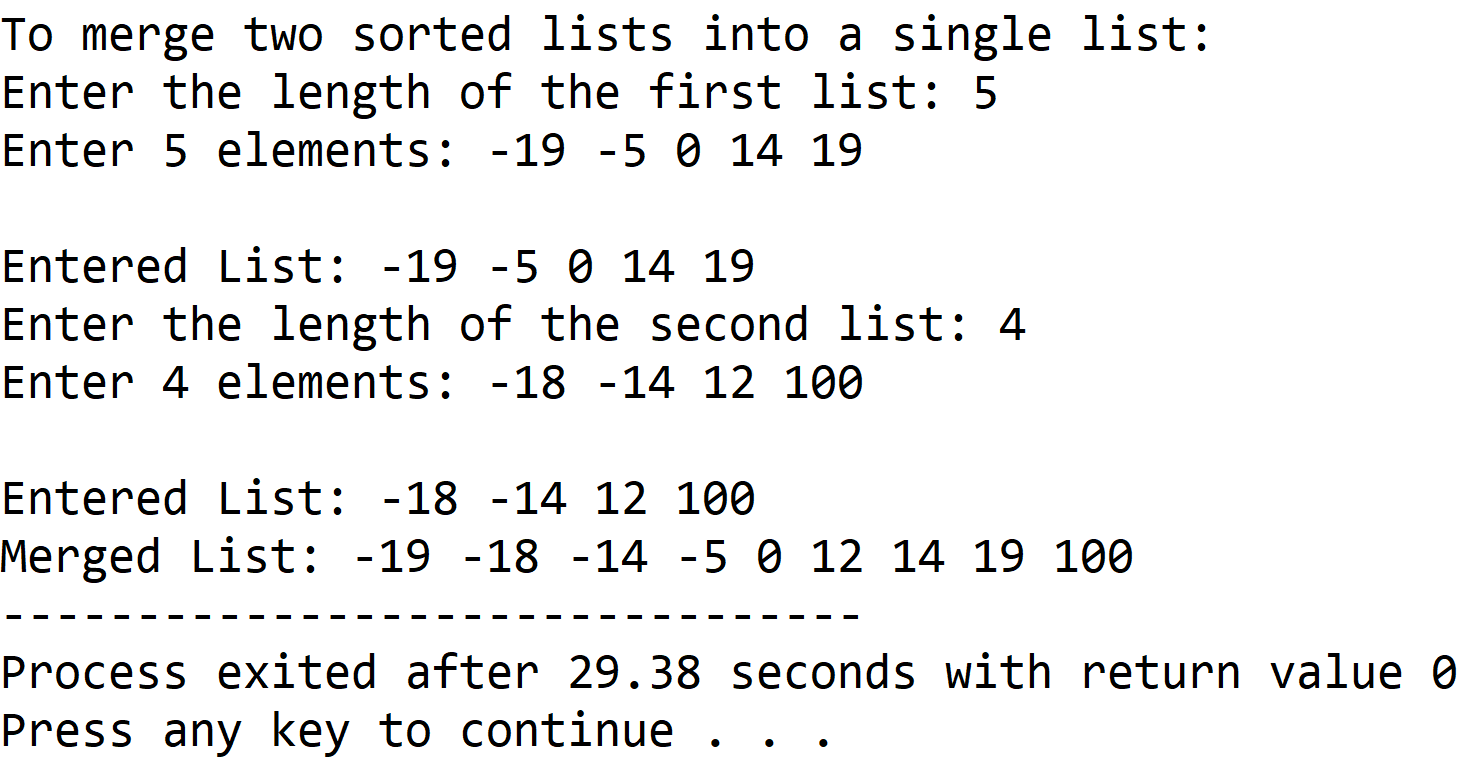
    displist(start3);

    return 0;

}

**4.OUTPUT**

**SET 1:** Merging of two sorted lists



**5.DISCUSSIONS**

**Variable Description**

* **start1,start2,start3:** header nodes of the first, second and third lists repectively.
* **len1,len2,len3:**Number of elements in first,second and third list respectively.
* **temp1,temp2,ptr:** pointers to nodes.
* **Item:** to hold user entered data during insertion of nodes.
* **I,j,k:** loop counters.

**Limitations:**

* Unlike arrays, the nodes need extra memory locations as pointers to refer to other nodes in the linked list, this leads to more usage of memory.
* There is no intrinsic indexing of the nodes, thus iterative constructs have to be used to find the location of a node.
* There is no provision to traverse backwards in the list.

**Uses**

* The program can be used to merge any two sorted list of integers into a single sorted list. It is of huge importance in several sorting procedures such as merge sort.

**Future Scope**

* A doubly linked list can be used to add the provision of backwards traversal in the list
* A extra index field can be added to each node to indicate the location of each node.