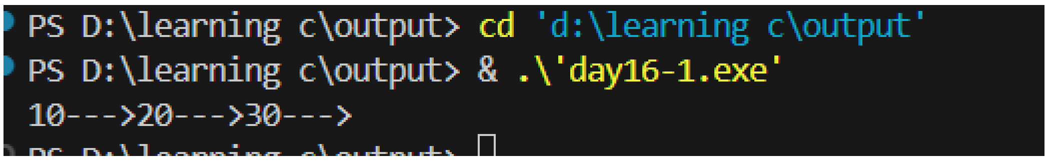
DAY 16

```
//linkedlist
/* 1.create a structure representation of linked node in c
struct Node
{ //data fileds
  int data;
  //pointer field(points to next node)
  struct Node *next;
 };
 2.creating a node for a linked list (dynamic way)
  struct Node *node1=(struct Node *)malloc(sizeof(struct Node))
  3.shortening the node declaration
 typedef struct Node
 { //data fileds
  int data;
  //pointer field(points to next node)
  struct Node *next;
 }Node;
 Node *node1=(Node *)malloc(sizeof(Node));
 4.assigning values to member elements of the node
 node1->data=10;
 node1->next=NULL;//here since only one node present

*/
1.  //linkedlist
/* 1.create a structure9representation of linked node in c
struct Node
{ //data fileds
  int data;
  //pointer field(points to next node)
  struct Node *next;
 };
 2.creating a node for a linked list (dynamic way)
  struct Node *node1=(struct Node *)malloc(sizeof(struct Node))
  3.shortening the node declaration
 typedef struct Node
 { //data fileds
  int data;
  //pointer field(points to next node)
  struct Node *next;
 }Node;
 Node *node1=(Node *)malloc(sizeof(Node));
 4.assigning values to member elements of the node
 node1->data=10;
 node1->next=NULL;//here since only one node present

*/
#include<stdio.h>
#include<stdlib.h>
//define the structure of the node
typedef struct Node
{ //data fileds
  int data;
  //pointer field(points to next node)
  struct Node *next;
}Node;
int main()
{
   //creating the first node
    Node *node1=(Node *)malloc(sizeof(Node));
    //assigning values
    node1->data=10;
```

```
// node1->next=NULL;
//creating the second node
 Node *node2=(Node *)malloc(sizeof(Node));
 //assigning values
 node2->data=20;
 //creating the third node
 Node *node3=(Node *)malloc(sizeof(Node));
 //assigning values
 node3->data=30;
 //now linking of nodes
 //first node to second , second node to third
 node1->next=node2;
 node2->next=node3;
 node3->next=NULL;
 //printf("%d %p",node1->data,node1->next);
 //printing the linked list
 //1.traverse from first to third
   //a.create a temp pointer of type struct Node
   //b.point to first node , make temp pointer point to first node
   //c.move the temp pointer from first to third node for printing entire linked list
   //use a loop , till node points to null
   //ie till loop!=NULL
Node *temp;
temp=node1;
while(temp!=NULL)
{
    printf("%d--->",temp->data);
    temp=temp->next;
}
```

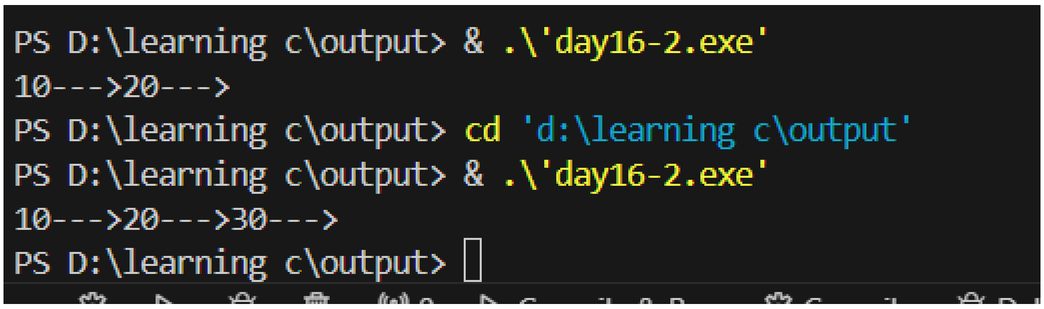
```
    return 0;

}
```



```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day16-1.exe'
10--->20--->30--->
```

2. //using function to create node

```
#include<stdio.h>

#include<stdlib.h>

typedef struct Node

{

   int data;

   struct Node *next;

}Node;

//node is going to return adress so pointer to struct node should be the type of function

Node* createNode(int data);

int main()

{

   //10 pointing to null

   Node *first=createNode(10);

   //10 pointing to 20 , 20 pointing to null

   first->next=createNode(20);

   //10 ->20->30->NULL

   first->next->next=createNode(30);


   Node *temp;

   temp=first;

   while(temp!=NULL)

   {

      printf("%d--->",temp->data);

      temp=temp->next;
   }
  return 0;
}
Node* createNode(int data)
{
   Node *newnode=(Node *)malloc(sizeof(Node));
   newnode->data=data;
   //intially assigning next field of newly created node to null
   newnode->next=NULL;
   return newnode;

}
```



```
PS D:\learning c\output> & .\'day16-2.exe'
10--->20--->
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day16-2.exe'
10--->20--->30--->
PS D:\learning c\output> 
```

3. create a node in a linked list which will have the following details of student Name, roll number, class, section, an array having marks of any three subjects Create a linked list for 5 students and print it.

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

typedef struct Student {
   char name[10];
   int rollnumber;
   char section;
   int marks[3];
   struct Student *next;
```

```
} Student;

Student *createnode(char[], int, char, int[]);

int main() {
    int n = 5;  // Number of students
    Student *first = NULL, *last = NULL;

    for (int i = 0; i < n; i++) {
        Student std;

        printf("Enter the name of student: ");
        scanf("%s", std.name);
        printf("Enter the roll number: ");
        scanf("%d", &std.rollnumber);
        printf("Enter the section: ");
        scanf(" %c", &std.section);  // Note the space before %c
        printf("Enter marks of three subjects:\n");
        for (int j = 0; j < 3; j++) {
            printf("Subject %d: ", j + 1);
            scanf("%d", &std.marks[j]);
        }

        // Create a new node for the student
        Student *new_node = createnode(std.name, std.rollnumber, std.section, std.marks);

        // Append the new node to the linked list
        if (first == NULL) {
            first = new_node;
            last = new_node;
        } else {
            last->next = new_node;
            last = new_node;
        }
    }

    // Print the student information
    Student *temp = first;
    while (temp != NULL) {
        printf("\nStudent Information:\n");
        printf("Name: %s\n", temp->name);
        printf("Roll Number: %d\n", temp->rollnumber);
        printf("Section: %c\n", temp->section);
        printf("Marks: %d, %d, %d\n", temp->marks[0], temp->marks[1], temp->marks[2]);
        temp = temp->next;
    }

    return 0;
}

Student *createnode(char name[10], int rollnumber, char section, int marks[3]) {
    // Creating a node
    Student *stnode = (Student *)malloc(sizeof(Student));
    strcpy(stnode->name, name);
    stnode->rollnumber = rollnumber;
    stnode->section = section;
    for (int i = 0; i < 3; i++) {
        stnode->marks[i] = marks[i];
    }
    stnode->next = NULL;
    return stnode;
}
```

```
PS D:\learning c> cd 'd:\learning c\output'

PS D:\learning c\output> & .\'day16-3.exe'

Enter the name of student: rinta

Enter the roll number: 1

Enter the section: a

Enter marks of three subjects:

Subject 1: 78

Subject 2: 90

Subject 3: 45

Enter the name of student: rani

Enter the roll number: 2

Enter the section: a

Enter marks of three subjects:

Subject 1: 78

Subject 2: 89

Subject 3: 90

Enter the name of student: raju

Enter the roll number: 3

Enter the section: a

Enter marks of three subjects:

Subject 1: 56

Subject 2: 78

Subject 3: 90

Enter the name of student: ria

Enter the roll number: 4

Enter the section: a

Enter marks of three subjects:

Subject 1: 67

Subject 2: 78
Subject 3: 90

Enter the name of student: richa

Enter the roll number: 5

Enter the section: a

Enter marks of three subjects:

Subject 1: 67

Subject 2: 56

Subject 3: 99


Student Information:

Name: rinta

Roll Number: 1

Section: a

Marks: 78, 90, 45


Student Information:

Name: rani

Roll Number: 2

Section: a

Marks: 78, 89, 90


Student Information:

Name: raju

Roll Number: 3

Section: a

Marks: 56, 78, 90


Student Information:

Name: ria

Roll Number: 4

Section: a
```

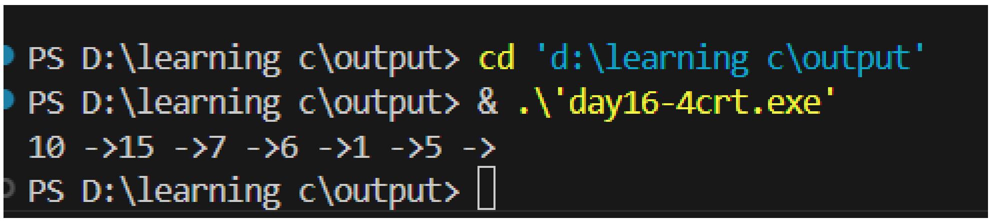
```
Marks: 67, 78, 90

Student Information:
Name: richa
Roll Number: 5
Section: a
Marks: 67, 56, 99
PS D:\learning c\output>
```

4.
```c
#include <stdio.h>
#include <stdlib.h>

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

```
}Node;

//Function with dual purpose: Creating a new node also adding a new node at the beginning
void InsertFront(Node** ,int );
void InsertMiddlle(Node* , int);
//Function with dual purpose: Creating a new node also adding a new node at the end
void InsertEnd(Node**, int);
void printList(Node*);

int main(){
   Node* head = NULL;
   InsertEnd(&head, 6);
   InsertEnd(&head, 1);
   InsertEnd(&head, 5);
   InsertFront(&head, 7 );
   InsertFront(&head, 10 );
   InsertMiddlle(head,15);
   printList(head);

   return 0;
}

void InsertEnd(Node** ptrHead, int nData){
   //1.Creating a Node
   Node* new_node=(Node *)malloc(sizeof(Node));
   //1.1 Create one more pointer which will point to the last element of the linked list
   Node* ptrTail;
   ptrTail = *ptrHead;
   //2.Enter nData
   new_node->data = nData;
   //3. we have to make the next field as NULL
```

```
    new_node->next = NULL;
    //4. If the linked list is empty make ptrHead point to thge new node created
    if(*ptrHead == NULL){
        *ptrHead = new_node;
        return;
    }
    //5. else Traverse till the last node and insert the new node at the end
    while(ptrTail->next != NULL){
        //5.1 MOve the ptrTail pinter till the end
        ptrTail = ptrTail->next;
    }
    ptrTail->next = new_node;
return;
}

void InsertFront(Node** ptrHead,int nData){
     //1. Create a New Node
     Node* new_node = (Node*)malloc(sizeof(Node));
     //2. Assign Data to the new Node
     new_node->data = nData;
     //3. Make the new node point to the first node of the linked list
     new_node->next = (*ptrHead);
     //4. Assign a the address of new Node to ptrHead
    (*ptrHead) = new_node;
}
void InsertMiddlle(Node* head, int nData)
{//1.check if list is empty
   if(head==NULL ||head->next==NULL)
   {
    printf("list empty \n");
    return;
```

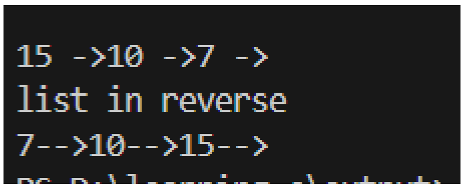
```
  }
  //2.create new node
   Node* new_node = (Node*)malloc(sizeof(Node));
   new_node->data=nData;
   //3.inserting new node between two nodes
   new_node->next=head->next;
   head->next=new_node;

}

void printList(Node* node){
   while (node != NULL){
      printf("%d ->",node->data);
      node = node->next;
   }
}
```



```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day16-4crt.exe'
10 ->15 ->7 ->6 ->1 ->5 ->
PS D:\learning c\output>
```

5.

```
//reverse a linked list
//requirements
//1.define a function to reverse the linkedlist iteratively
//2.update the head pointer to the new firstnode
//3.display the reversed lidt

#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int data;

    struct node *next;

}Node;

void insertdata(Node **,int);

void reverselist(Node *);

int main()

{

    //1.insert data to the linked list

    //2.reverse the linked list


    Node *head=NULL;//inthe begining header is null since no elements


    insertdata(&head, 7 );

    insertdata(&head, 10 );

    insertdata(&head,15);

    printList(head);

    printf("list in reverse \n");

    reverselist(head);





}

void insertdata(Node ** ptrHead,int ndata)

{

    //creating the nodes to create the linked list

      Node* new_node = (Node*)malloc(sizeof(Node));//created a new node

      //now we have to insert data to this node

      new_node->data=ndata;

       //3. Make the new node point to the first node of the linked list

       new_node->next = (*ptrHead);

     //4. Assign a the address of new Node to ptrHead
      (*ptrHead) = new_node;



}
void reverselist(Node * head)
{
    //we have to traverse from the end to begining
    //tail is at end ie ==null from there to first
    if(head==NULL)
    {
       //that means end of the list
       return;
    }
    reverselist(head->next);
    printf("%d-->",head->data);

}

void printList(Node* node){
    while (node != NULL){
       printf("%d ->",node->data);
       node = node->next;
    }
}
```



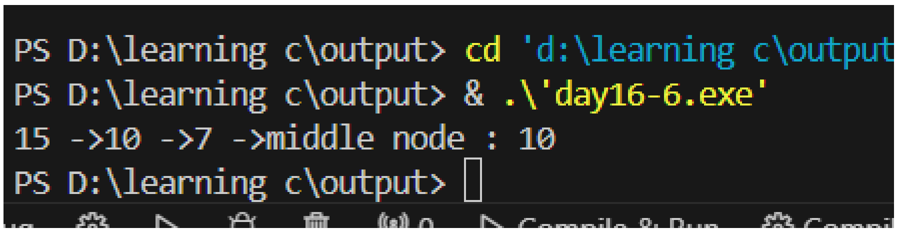
```
15 ->10 ->7 ->
list in reverse
7-->10-->15-->
```

```
6.  //find the middle node
//1.use two pointers:one moving one step and other moving two steps
//2.when the faster pointer reaches the end , the slower pointer will point to the middle node
//if the list has even nodes , display the first middle node
```

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
   int data;
   struct node *next;
}Node;
void insertdata(Node **,int);
void printList(Node* );
void displaymiddle(Node *);
int main()
{
   //1.insert data to the linked list
   //2.reverse the linked list

   Node *head=NULL;//inthe begining header is null since no elements

   insertdata(&head, 7 );
   insertdata(&head, 10 );
   insertdata(&head,15);
   printList(head);
   displaymiddle(head);





}
void insertdata(Node ** ptrHead,int ndata)
{
   //creating the nodes to create the linked list
```

```
     Node* new_node = (Node*)malloc(sizeof(Node));//created a new node
     //now we have to insert data to this node
     new_node->data=ndata;
      //3. Make the new node point to the first node of the linked list
      new_node->next = (*ptrHead);
    //4. Assign a the address of new Node to ptrHead
     (*ptrHead) = new_node;



}
void displaymiddle(Node *head)
{
   if(head==NULL)
   {
      printf("list is empty \n");
      return;
   }
   Node *slow=head;//pointer moving one step at a time
   Node *fast=head;//pointer moving two step at a time
   while(fast!=NULL && fast->next!=NULL)
   {
      slow=slow->next;
      fast=fast->next->next;
   }
   printf("middle node : %d \n",slow->data);
}



void printList(Node* node){
   while (node != NULL){
      printf("%d ->",node->data);
```

```
        node = node->next;
    }
}
```



```
PS D:\learning c\output> cd 'd:\learning c\output
PS D:\learning c\output> & .\'day16-6.exe'
15 ->10 ->7 ->middle node : 10
PS D:\learning c\output> 
```