LABORATORUL 5

1.Implementati o structura de lista dublu inlantuita (contine un camp de date si doi 2 pointeri, unul spre nodul urmator si unul catre nodul anterior). Adaugati functionalitatea de : - Crearea unui nodul - Adaugarea la inceputul listei - Adaugare la sfarsitul listei - Printare - Stergere de memorie

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
  int data;
  struct node* next;
  struct node* prev;
} node;

node* create(int data,node* next, node* prev)
{
  node* new_node = (node*)malloc(sizeof(node));
  new_node->data = data;
  new_node->prev = prev;
  new_node->next = next;

  return new_node;
}

node* prepend(node* head,int data)
{
  node* new_node = create(data,NULL,NULL);
  if(head == NULL)
  {
     head = create(data,NULL,NULL);
     return head;
}
  head->prev = new_node;
  new_node->next = head;
  head = new_node;

  return head;
}

node* append(node* head,int data)
{
  node* new_node = create(data,NULL,NULL);
  node* cursor = head;
  if(head == NULL)
  {
    head = new_node;
    return head;
  }

 while(cursor->next != NULL) cursor=cursor->next;
 cursor->next = new_node;
 new_node->prev = cursor;

 return head;
}

void print(node* head)
{
  node* cursor = head;
  while(cursor!=NULL)
  {
    printf("%d",cursor->data);
    cursor = cursor->next;
  }
}


void dispose(node* head)
{
  printf("in function dispose");
   node* cursor,*tmp;
    if(head !=NULL)
     {
        if(head->prev != NULL)
{
          head->prev->next = NULL;
          dispose(head->prev);

}       if(head->next != NULL)
{
         head->next->prev = NULL;
         dispose(head->next);
}
        //dispose(head->prev);
        //head->next=NULL;
        printf("Deleting %d\n",head->data);
        free(head);
     }
```

```c
}

int main()
{

  node* head = NULL;

  head = create(1,NULL,NULL);

 head = prepend(head,4);

 print(head);
 printf("\n");
 head = append(head,3);
 head = prepend(head,6);
 print(head);
 printf("\n");
 dispose(head->next);
 print(head);
 return 0;
}
```