```cpp
#include <iostream>

using namespace std ;

struct element
{
    int data ;
    struct element * next ;
    struct element * prev ;
};

typedef struct element Element ;
typedef Element *       DList   ;


void display_list(DList lst)
{
    DList ptr = lst ;

    if(lst == NULL)
    {
        cout << "Empty list." << endl ;

        return ;
    }

    while(ptr != NULL)
    {
        cout << ptr->data << " " ;

        ptr = ptr->next ;
    }
    cout << endl ;

    return ;
}

bool node_exist(DList lst, DList pos)
{
    DList ptr = lst ;

    while(ptr != NULL)
    {
        if(ptr == pos)
            return true ;

        ptr = ptr->next ;
    }

    return false ;
}

DList insert_node(DList lst, int value, DList pos, bool &succ)
{
    succ = true ;

    if(!node_exist(lst, pos)&&(pos != NULL))
    {
        succ = false ;

        return lst ;
    }

    DList ptr = new Element ; // creating the node

    ptr->data = value ;          // filling the node fields
    ptr->prev = NULL  ;
    ptr->next = NULL  ;

    if(lst == NULL)  // List empty, the node becomes the list
    {
        return ptr ;
    }
    else
    {
        if(pos == lst)    // Insert in the first position
```

```
        {
            ptr->next  =  lst  ;
            lst->prev  =  ptr  ;
            lst        =  ptr  ;

            return  lst  ;
        }
        else
        {
            if((pos == NULL)||(pos->next == NULL))   // Insert in the last position
            {
                DList  pLast  =  lst  ;

                while(pLast->next != NULL)    // looking for the last node
                {
                    pLast  =  pLast->next  ;
                }

                pLast->next  =  ptr    ;  // connecting the last node to the new node
                ptr->prev    =  pLast  ;  // connecting the new node to the lest node

                return  lst  ;
            }
            else  // Inserting the new node in the middle of the list
            {
                // connecting the node following the insertion position to the new node
                pos->next->prev  =  ptr  ;
                // connecting the new node to the node following the insertion position
                ptr->next  =  pos->next  ;

                // connecting the node of the insertion position to the new node
                pos->next  =  ptr        ;
                // connecting the new node to the node of the insertion position
                ptr->prev  =  pos  ;

                return  lst  ;
            }
        }
    }
}

DList  delete_node(DList lst, DList pos, bool &succ)
{
    succ  =  true  ;

    if((lst == NULL)||!node_exist(lst, pos))
    {
        succ  =  false  ;
        return   lst    ;
    }

    if(pos->next != NULL)
        pos->next->prev  =  pos->prev  ;

    if(pos->prev != NULL)
        pos->prev->next  =  pos->next  ;
    else
        lst  =  lst->next  ;

    delete  pos  ;

    return  lst  ;
}

int  main()
{
    DList  lst   =  NULL   ;

    bool   test  =  false  ;

//    DList  ptr   =  new  Element   ;

    lst  =  insert_node(lst, 5, NULL, test)  ;
    cout  <<  "insertion in an empty list ? : "  <<  test  <<  endl  ;
    display_list(lst)  ;

```

```
    lst  =  delete_node(lst, lst, test)  ;
    cout  <<  "deletion of one element list ? : "  <<  test  <<  endl  ;
    display_list(lst)  ;

    cout  <<  "made a small list by inserting in the beginning :"  <<  endl  ;
    for(int i = 1 ; i <= 4 ; i++)
        lst  =  insert_node(lst, i, lst, test)  ;
    display_list(lst)  ;

//    DList  ptr  =  lst  ;
//    while(ptr->next != NULL)
//    {
//        ptr  =  ptr->next  ;
//    }
    lst  =  insert_node(lst, 7, NULL, test)  ;
    cout  <<  "insertion in the end of the list ? : "  <<  test  <<  endl  ;
    display_list(lst)  ;

    lst  =  insert_node(lst, 10, lst->next, test)  ;
    cout  <<  "insertion in the middle of the list ? : "  <<  test  <<  endl  ;
    display_list(lst)  ;

    lst  =  delete_node(lst, lst, test)  ;
    cout  <<  "deletion in the beginning of a list ? : "  <<  test  <<  endl  ;
    display_list(lst)  ;

    lst  =  delete_node(lst, lst->next, test)  ;
    cout  <<  "deletion in the middle of a list ? : "  <<  test  <<  endl  ;
    display_list(lst)  ;

    lst  =  delete_node(lst, lst->next->next->next, test)  ;
    cout  <<  "deletion in the end of a list ? : "  <<  test  <<  endl  ;
    display_list(lst)  ;

    return  0 ;
}
```