```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
#include<ctype.h>
#include<conio.h>

    //protip: The only possible search algorithm is Linear, because Linked list is daisy chained.
    //to make any other algorithm which is non-linear possible, declare temporary array and insert those data from the linked list there.
    //then you can do something with that search target. yeah, only as assisting tool.
    //don't forget clean the array after use!

struct tnode { //double linked list
    int angka;

    tnode *next;
    tnode *prev;
}*head = NULL, *tail = NULL, *temp=NULL;

void pushHead(int angka){
    tnode *nd = (tnode *) malloc(sizeof(tnode));

    nd->angka = angka;

    if(head==NULL){
        head=tail=nd;
    } else {
        head->prev = nd; //Forward arrow
        nd->next = head; //Backward arrow, remember to connect back to it's predecessor
        head = nd;
    }
    tail->next = NULL;
    head->prev = NULL;
}

void pushTail(int angka){
    tnode *nd = (tnode *) malloc(sizeof(tnode));

    nd->angka = angka;

    if(head==NULL){
        head=tail=nd;
    } else {
        tail->next = nd; //Forward arrow
        nd->prev = tail; //Backward arrow, remember to connect back to it's predecessor
        tail = nd;
    }
    tail->next = NULL;
    head->prev = NULL;
}

```

```
52  void popHead(){ //erase first data from head
53      if(head){ //head != NULL
54          //if there is data
55          if(head==tail){
56              //if data is only consists 1 set
57              free(head); //or tail, whatever.
58              head = tail = NULL;
59          } else {
60              head = head->next;
61              free(head->prev); //big advantage of double linked list, no need
                  temp. reduces data loss
62              head->prev = NULL; //reassign deleted data as NULL
63          }
64      } else {
65          //if there is no data
66          printf("tidak ada data");
67      }
68  }
69
70  void popTail(){ //erase first data from tail
71      if(head){ //head != NULL
72          //if there is data
73          if(tail==head){
74              //if data is only consists 1 set
75              free(tail); //or head, whatever.
76              head = tail = NULL;
77          } else {
78              tail = tail->prev;
79              free(tail->next); //big advantage of double linked list, no need
                  temp. reduces data loss
80              tail->next = NULL; //reassign deleted data as NULL
81          }
82      } else {
83          //if there is no data
84          printf("tidak ada data");
85      }
86  }
87
88  void popRefference(int angka){
89      int flag = 0;
90      temp = head;
91      if(head){
92          if(head->angka == angka){ //edge on Head
93
94              /*if(head==tail){//if data is left 1 on the list
95                  free(head);
96                  head = tail = NULL;
97              } else{//if data > 1 on the list
98                  head = head->next;
99                  free(head->prev); //big advantage of double linked list, no
                      need temp. reduces data loss
100                 head->prev = NULL; //reassign deleted data as NULL
101             }*/
102
103             popHead();
104
```

```
105         } else if(tail->angka == angka){ //edge on Tail
106             popTail();
107 
108         } else { //in-between
109             temp = head;
110             while(temp != NULL && temp -> angka != angka){
111                 temp = temp ->next;
112             }
113             if(temp!=NULL){//if data being search is found
114                 tnode *temp2 = head;
115                 while(temp2->next != temp){
116                     temp2= temp2 ->next;
117                 }
118                 temp2->next = temp->next;
119                 free(temp);
120             }
121         }
122 
123     } else {
124         printf("tidak ada data");
125     }
126 }
127 
128 void popSearch(int angka){ //this one is better
129     if(head){
130         temp = head;
131 
132         while(temp!=NULL && temp->angka!=angka){
133             temp = temp->next;
134         }
135         if(temp==NULL){
136             printf("angka tidak ditemukan");
137         } else {
138             if(temp==head)popHead();
139             else if(temp==tail)popTail();
140             else {
141                 temp->prev->next = temp->next;
142                 temp->next->prev = temp->prev;
143                 free(temp);
144             }
145         }
146     }
147 }
148 
149 void popAll(){
150     while(head != NULL) {
151         popHead();
152     }
153 }
154 
155 void view(){
156     temp = head;
157     printf("NULL<->");
158     while(temp != NULL){
159         printf("%d<->", temp->angka);
160         temp = temp->next;
```

```
161     }
162     printf("NULL");
163 }
164
165 int main() {
166     int win;
167
168     /*pushHead(5);
169     pushHead(4);
170     pushHead(10);
171     pushTail(111);*/
172
173     pushTail(6);
174     pushTail(7);
175     pushTail(8);
176     pushTail(9);
177     pushTail(10);
178
179     //popHead();
180     //popTail();
181
182     /*popRefference(6);
183     popRefference(7);
184     popRefference(8);
185     popRefference(9);
186     popRefference(10);*/
187     //popHead();
188
189     //popSearch(10);
190     //popSearch(8);
191     //popSearch(1009);
192     //popSearch(11);
193
194     popAll();
195
196     view();
197
198     getchar();
199     return 0;
200 }
```