Program to

Insert(5types)

Delete(4types)

Search

Display

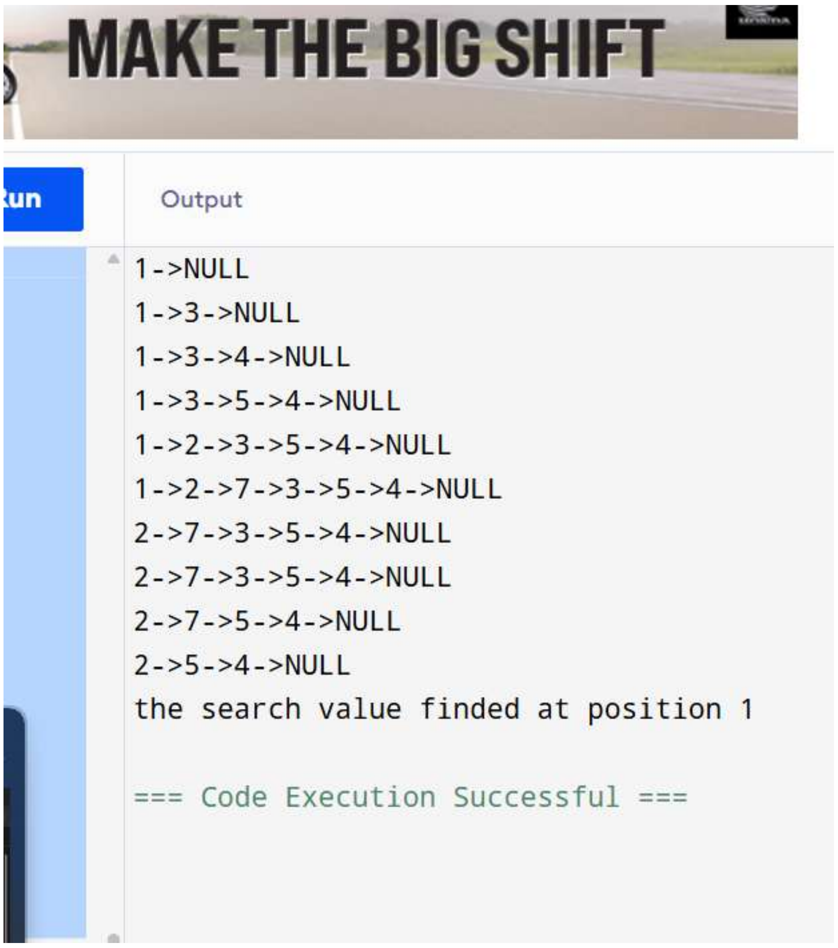
From a singly linked list

Program

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
   int data;
   struct Node* next;
};
struct Node* head=NULL;

struct Node* createNode(int data){
   struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
   newNode->data=data;
   return newNode;
}
void insertatbegin(int data){
   struct Node* newNode=createNode(data);
   if(head==NULL){
      head=newNode;
   }
   else{
      newNode->next=head;
      head=newNode;
   }
}
void insertatend(int data){
    struct Node* newNode=createNode(data);
    if(head==NULL){
        head=newNode;
    }
    else{
        struct Node* temp=head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        newNode->next=temp->next;
        temp->next=newNode;
    }
}
void insertaftervalue(int value,int data){
    struct Node* newNode=createNode(data);
    struct Node* temp=head;
    while(temp!=NULL&&temp->data!=value){
        temp=temp->next;
    }
    if(temp==NULL){
        printf("value not found");
        free(newNode);
    }
    else{
        newNode->next=temp->next;
        temp->next=newNode;
    }
}
void insertbeforevalue(int value,int data){
    struct Node* newNode=createNode(data);
    if(head->data==value){
        newNode->next=head;
        head=newNode;
    }
    else{
        struct Node* temp=head;
        struct Node* prev=NULL;
        while(temp!=NULL&&temp->data!=value){
            prev=temp;
            temp=temp->next;
        }
        newNode->next=prev->next;
        prev->next=newNode;

    }
}
void insertatindex(int index,int data){
    struct Node* newNode=createNode(data);
    int i=1;
    if(index==0){
        newNode->next=head;
        head=newNode;
    }
    else{
        struct Node* temp=head;
        while(temp!=NULL&&i<index-1){
            temp=temp->next;
            i++;
        }
        newNode->next=temp->next;
        temp->next=newNode;
    }
```

```
}
void dltatbegin(){
   if(head==NULL){
      printf("no elements to dlt");
   }
   else{
      struct Node* temp=head;
      head=temp->next;
      free(temp);
   }
}
void dltatend(){
   struct Node* temp=head;
   while(temp->next!=NULL){
      temp=temp->next;
   }

   free(temp->next);
   temp->next=NULL;
}
void dltbyvalue(int data){
   struct Node* temp=head;
   while(temp->next!=NULL&&temp->next->data!=data){
      temp=temp->next;
   }
   struct Node* dltn=temp->next;
   temp->next=dltn->next;
   free(dltn);

}
void dltbyindex(int index){
    int i=1;
    struct Node* temp=head;
    if(index==1){
        head=temp->next;
        free(head);
    }
    else{
        while(temp->next!=NULL&&i<index-1){
            temp=temp->next;
            i++;
        }
        struct Node* dltn=temp->next;
        temp->next=dltn->next;
        free(dltn);
    }

}
void search(int data){
    struct Node* temp=head;
    int i=1;
    while(temp!=NULL&&temp->data!=data){
        temp=temp->next;
        i++;
    }
    printf("the search value finded at position %d",i);
}

void display(){
    if(head==NULL){
        printf("the list is empty");
    }
    else{
        struct Node* temp=head;
        while(temp!=NULL){
            printf("%d->",temp->data);
            temp=temp->next;
        }
       printf("NULL\n");
    }
}
int main(){
    insertatbegin(1);
    display();
    insertatend(3);
    display();
    insertaftervalue(3,4);
    display();
     insertaftervalue(3,5);
    display();
    insertbeforevalue(3,2);
    display();
    insertatindex(3,7);
    display();
    dltatbegin();
    display();
    dltatend();
    display();
    dltbyvalue(3);
    display();
    dltbyindex(2);
    display();
    search(2);
    return 0;

}
```

Output



```
1->NULL
1->3->NULL
1->3->4->NULL
1->3->5->4->NULL
1->2->3->5->4->NULL
1->2->7->3->5->4->NULL
2->7->3->5->4->NULL
2->7->3->5->4->NULL
2->7->5->4->NULL
2->5->4->NULL
the search value finded at position 1

=== Code Execution Successful ===
```