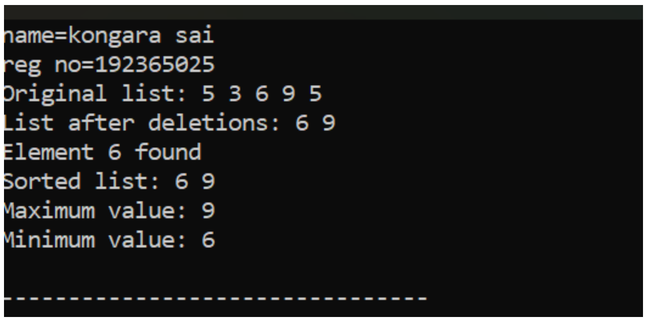
Day-4

Tuesday, 5 March 2024 9:33 pm

Doubly linked list

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

struct node {
   int data;
   struct node *prev;
   struct node *next;
}*n,*head ,*tail;


 struct node *createNode(int data) {
   n= (struct node*)malloc(sizeof(struct node));
   if (n == NULL) {
      printf("Memory allocation failed\n");
      exit(1);
   }
   n->data = data;
   n->prev = NULL;
   n->next = NULL;
   return n;
}

 void insertBeg(int data) {
   n = createNode(data);
   if (head == NULL) {
      head = tail = n;
   } else {
      n->next = head;
      head->prev = n;
      head = n;
   }
}

 void insertEnd(int data) {
   n = createNode(data);
   if (head == NULL) {
      head = tail = n;
   } else {
      tail->next = n;
      n->prev = tail;
      tail = n;
   }
}

 void insertMid(int data, int mid_data) {
   struct node *t = head;
   while (t != NULL) {
      if (t->data == mid_data) {
         n= createNode(data);
```

```
            n->prev = t;
            n->next = t->next;
            if (t->next != NULL) {
                t->next->prev = n;
            } else {
                tail = n;
            }
            t->next = n;
            break;
        }
        t = t->next;
    }
}

 void deleteBeg() {
    if (head == NULL) {
        return;
    }
    struct node *t = head;
    head = head->next;
    if (head != NULL) {
        head->prev = NULL;
    } else {
        tail = NULL;
    }
    free(t);
}

 void deleteEnd() {
    if (head == NULL) {
        return;
    }
    struct node *t = tail;
    tail = tail->prev;
    if (tail != NULL) {
        tail->next = NULL;
    } else {
        head = NULL;
    }
    free(t);
}

void deleteMid(int mid_data) {
    struct node *t = head;
    while (t != NULL) {
        if (t->data == mid_data) {
            if (t == head) {
                deleteBeg();
            } else if (t == tail) {
                deleteEnd();
            } else {
                t->prev->next = t->next;
                t->next->prev = t->prev;
                free(t);
            }
            break;
        }
        t = t->next;
    }
}

void display() {
    struct node *t = head;
    while (t != NULL) {
        printf("%d ", t->data);
        t = t->next;
    }
    printf("\n");
}

void search(int key) {
    struct node *t = head;
    while (t != NULL) {
        if (t->data == key) {

        }
        t = t->next;
    }

}

void sort() {
    struct node *current = head, *index = NULL;
    int temp;
    while (current != NULL) {
        index = current->next;
        while (index != NULL) {
            if (current->data > index->data) {
                temp = current->data;
                current->data = index->data;
                index->data = temp;
            }
            index = index->next;
        }
        current = current->next;
    }
}

int findMax() {
    int max = head->data;
    struct node *temp = head->next;
    while (temp != NULL) {
        if (temp->data > max) {
            max = temp->data;
        }
        temp = temp->next;
    }
    return max;
}

int findMin() {
    int min = head->data;
    struct node *temp = head->next;
    while (temp != NULL) {
        if (temp->data < min) {
            min = temp->data;
        }
        temp = temp->next;
    }
    return min;
}

 int main() {
                printf("name=kongara sai\n");
    printf("reg no=192365025\n");
    insertBeg(3);
    insertBeg(5);
    insertEnd(9);
    insertMid(6, 3);
    insertEnd(5);

    printf("Original list: ");
    display();

    deleteBeg();
    deleteEnd();
    deleteMid(3);

    printf("List after deletions: ");
    display();

    search(6);
    if (head != NULL) {
        printf("Element 6 found\n");
    } else {
        printf("Element 6 not found\n");
    }

    sort();
    printf("Sorted list: ");
    display();

    printf("Maximum value: %d\n", findMax());
    printf("Minimum value: %d\n", findMin());

    return 0;
}
```



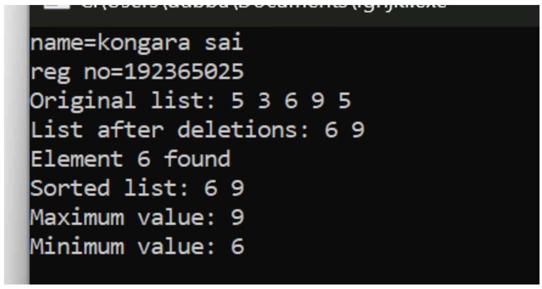
```
name=kongara sai
reg no=192365025
Original list: 5 3 6 9 5
List after deletions: 6 9
Element 6 found
Sorted list: 6 9
Maximum value: 9
Minimum value: 6

--------------------------------
```

Singly CLL

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

struct node {
    int data;
    struct node *next;
}*n;

struct node *head = NULL;
struct node *tail = NULL;

struct node *createNode(int data) {
  n = (struct node*)malloc(sizeof(struct node));
    if (n == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    n->data = data;
    n->next = NULL;
    return n;
}


void insertBeg(int data) {
    n = createNode(data);
    if (head == NULL) {
        head = tail = n;
        n->next = n;
    } else {
        n->next = head;
        head = n;
        tail->next = head;
    }
}


void insertEnd(int data) {
    n = createNode(data);
    if (head == NULL) {
        head = tail = n;
        n->next = n;
    } else {
        tail->next = n;
        tail = n;
        tail->next = head;
    }
}

 void insertMid(int data, int mid_data) {
    struct node *t = head;
    while (t != NULL) {
        if (t->data == mid_data) {
            n = createNode(data);
            n->next = t->next;
            t->next = n;
            if (t == tail) {
                tail = n;
            }
            break;
        }
        t = t->next;
    }
}

 void deleteBeg() {
    if (head == NULL) {
        return;
    }
    struct node *t = head;
    head = head->next;
    tail->next = head;
    free(t);
}


void deleteEnd() {
    if (head == NULL) {
        return;
    }
    struct node *t = head;
    while (t->next != tail) {
        t = t->next;
    }
    t->next = head;
    free(tail);
    tail = t;
}


void deleteMid(int mid_data) {
    struct node *prev = NULL;
    struct node *current = head;
    while (current != tail && current->data !=
mid_data) {
        prev = current;
        current = current->next;
    }
    if (current != NULL && current->data ==
mid_data) {
        if (current == head) {
            deleteBeg();
        } else if (current == tail) {
            deleteEnd();
        } else {
            prev->next = current->next;
            free(current);
        }
    }
}
```

```
void display() {
    struct node *t = head;
    if (t != NULL) {
      while (t != head) {
            printf("%d ", t->data);
            t = t->next;
        }
    }
    printf("\n");
}

 void search(int key) {
    struct node *t = head;
    if (t != NULL) {
         while (t != head) {
            if (t->data == key) {
                 exit(1);
            }
            t = t->next;
        }
    }

}


void sort() {
    struct node *current = head, *index = NULL;
    int t;
    if (head != NULL) {
        do {
            index = current->next;
            while (index != head) {
                if (current->data > index->data) {
                    t = current->data;
                    current->data = index->data;
                    index->data = t;
                }
                index = index->next;
            }
            current = current->next;
        } while (current != head);
    }
}


int findMax() {
    int max = head->data;
    struct node *t = head->next;
    while (t != head) {
        if (t->data > max) {
            max = t->data;
        }
        t = t->next;
    }
    return max;
}
```

```
int findMin() {
    int min = head->data;
    struct node *t = head->next;
    while (t != head) {
        if (t->data < min) {
            min = t->data;
        }
        t = t->next;
    }
    return min;
}

int main() {
    insertBeg(3);
    insertBeg(5);
    insertEnd(9);
    insertMid(6, 3);
    insertEnd(5);
    printf("name=kongara sai\n");
    printf("reg no=192365025\n");
    printf("Original list: ");
    display();

    deleteBeg();
    deleteEnd();
    deleteMid(3);

    printf("List after deletions: ");
    display();

     search(6);
    if (head != NULL) {
        printf("Element 6 found\n");
    } else {
        printf("Element 6 not found\n");
    }

    sort();
    printf("Sorted list: ");
    display();

    printf("Maximum value: %d\n", findMax());
    printf("Minimum value: %d\n", findMin());

    return 0;
}
```



```
name=kongara sai
reg no=192365025
Original list: 5 3 6 9 5
List after deletions: 6 9
Element 6 found
Sorted list: 6 9
Maximum value: 9
Minimum value: 6
```