Name: Mithlesh Yeole
Roll no. : B3-B3-59

DS LAB Practical 4B

Code:

```
        #include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node* next;
};
struct node* insertbegin(struct node* head, int v) {
struct node* ptr = (struct node*)malloc(sizeof(struct node));
if (ptr == NULL) {
printf("NO SPACE\n");
return head;
}
ptr->data = v;
ptr->next = head;
head = ptr;
return head;
}
struct node* insertend(struct node* head, int v) {
struct node* ptr = (struct node*)malloc(sizeof(struct node));
if (ptr == NULL) {

printf("NO SPACE\n");
return head;
}
ptr->data = v;
ptr->next = NULL;
if (head == NULL) {
return ptr;
}
struct node* temp = head;
while (temp->next != NULL) {
temp = temp->next;
}
temp->next = ptr;
return head;
```

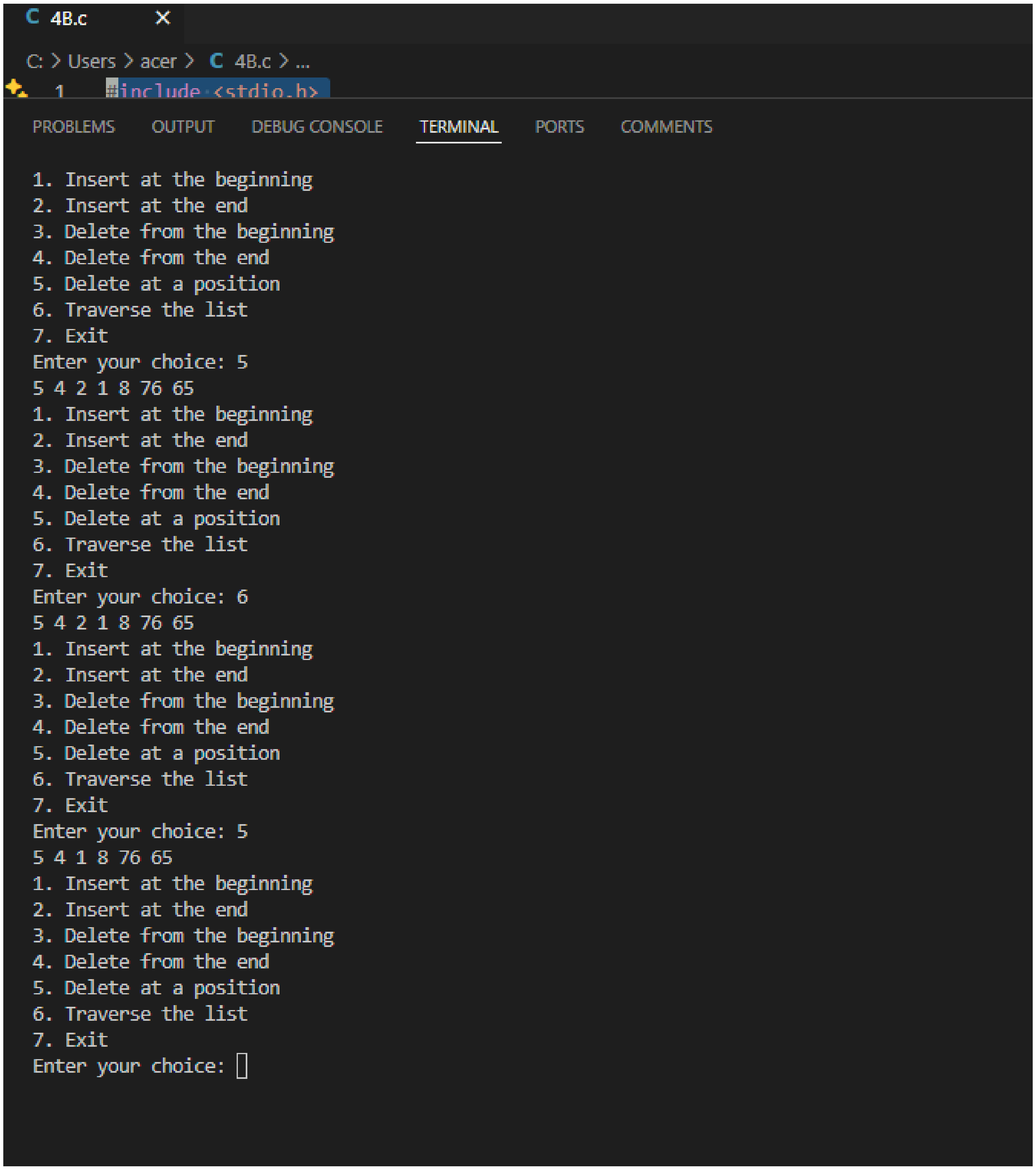
```c
}
struct node* insertAtPosition(struct node* head, int v, int pos) {
struct node* ptr = (struct node*)malloc(sizeof(struct node));
if (ptr == NULL) {
printf("NO SPACE\n");
return head;
}
ptr->data = v;
if (pos == 1) {
ptr->next = head;
return ptr;
}

struct node* temp = head;
for (int i = 1; i < pos - 1 && temp != NULL; i++) {
temp = temp->next;
}
if (temp == NULL) {
printf("Invalid position!\n");
free(ptr);
return head;
}
ptr->next = temp->next;
temp->next = ptr;
return head;
}

struct node* deletebegin(struct node* head) {

if (head == NULL) {
printf("List is empty!\n");
return head;
}
struct node* ptr = head;
head = head->next;
free(ptr);
return head;
}

struct node* deleteend(struct node* head) {
```

```c
if (head == NULL) {
printf("List is empty!\n");
return head;
}
if (head->next == NULL) {
free(head);
return NULL;
}
struct node* temp = head;
while (temp->next->next != NULL) {
temp = temp->next;
}
free(temp->next);
temp->next = NULL;
return head;

}
struct node* deleteAtPosition(struct node* head, int pos) {
    if (head == NULL) {
    printf("List is empty!\n");
    return head;
    }
    if (pos == 1) {
    return deletebegin(head);
    }
    struct node* temp = head;
    for (int i = 1; i < pos - 1 && temp != NULL; i++) {
    temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
    printf("Invalid position!\n");
    return head;
    }
    struct node* ptr = temp->next;
    temp->next = ptr->next;
    free(ptr);
    return head;
    }
```

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

int main() {
    struct node* head = NULL;
    struct node* newnode = NULL;
    int choice, data;
    while (1) {
        printf("1. Insert at the beginning\n");
        printf("2. Insert at the end\n");
        printf("3. Delete from the beginning\n");
        printf("4. Delete from the end\n");
        printf("5. Delete at a position\n");
        printf("6. Traverse the list\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
            printf("Enter data: ");
            scanf("%d", &data);
            newnode = (struct node*)malloc(sizeof(struct node));
            newnode->data = data;
            newnode->next = head;
            head = newnode;
            break;
            case 2:
            printf("Enter data: ");
            scanf("%d", &data);
            newnode = (struct node*)malloc(sizeof(struct node));
            newnode->data = data;
            newnode->next = NULL;
            if (head == NULL) {
                head = newnode;
            } else {
                struct node* temp = head;
                while (temp->next != NULL) {
                    temp = temp->next;
                }
                temp->next = newnode;
            }
            break;
            case 3:
                head = deletebegin(head);
            break;
            case 4:
                head = deleteend(head);
            break;
            case 5:
                head = deleteAtPosition(head,3);
            case 6:
                traverse(head);
            break;
            case 7:
                free(head);
            return 0;
            default:
            printf("Invalid choice!\n");
        }
    }
    for (int i = 0; i < 5; i++) {
        head = insertbegin(head, i);
        }

        printf("Linked List after inserting at the beginning:\n");
        traverse(head);
        head = insertend(head, 10);
        printf("Linked List after inserting at the end:\n");
        traverse(head);
        head = insertAtPosition(head, 99, 3);
        printf("Linked List after inserting 99 at position 3:\n");
        traverse(head);
        head = deletebegin(head);
        printf("Linked List after deleting from the beginning:\n");
```

```c
        traverse(head);
        head = deleteend(head);
        printf("Linked List after deleting from the end:\n");
        head = deleteAtPosition(head,3);
        printf("Linked list after deleting at position\n");
        traverse(head);

    return 0;
}
```

Output:


4B.c
C: > Users > acer > 4B.c > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

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

```
1. Insert at the beginning
2. Insert at the end
3. Delete from the beginning
4. Delete from the end
5. Delete at a position
6. Traverse the list
7. Exit
Enter your choice: 5
5 4 2 1 8 76 65
1. Insert at the beginning
2. Insert at the end
3. Delete from the beginning
4. Delete from the end
5. Delete at a position
6. Traverse the list
7. Exit
Enter your choice: 6
5 4 2 1 8 76 65
1. Insert at the beginning
2. Insert at the end
3. Delete from the beginning
4. Delete from the end
5. Delete at a position
6. Traverse the list
7. Exit
Enter your choice: 5
5 4 1 8 76 65
1. Insert at the beginning
2. Insert at the end
3. Delete from the beginning
4. Delete from the end
5. Delete at a position
6. Traverse the list
7. Exit
Enter your choice: 
```