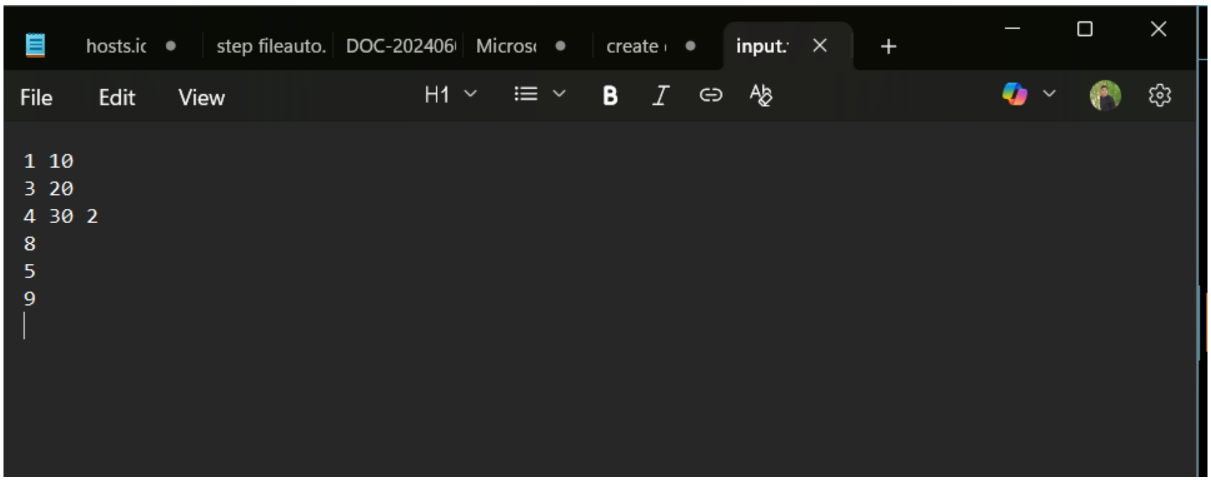
```c
#include <stdio.h>
#include <stdlib.h>


typedef struct Node {
   int data;
   struct Node *link;
} Node;

Node *head = NULL;
Node* createNODE(int data) {
   Node *nn = (Node *)malloc(sizeof(Node));
   if (!nn) {
      printf("Memory cant allocation");
      exit(0);
   }
   nn->data = data;
   nn->link = NULL;
   return nn;
}

void createList(int data) {
   Node *nn = createNODE(data);

   if (head == NULL) {
      head = nn;
      return;
   }

   Node *temp = head;
   while (temp->link != NULL)
      temp = temp->link;

   temp->link = nn;
}

void insertAtFirst(int data) {
   Node *nn = createNODE(data);
   nn->link = head;
   head = nn;
}

void insertAtEnd(int data) {
   createList(data);
}

void insertAtPosition(int data, int pos) {
   if (pos == 1) {
        insertAtFirst(data);
        return;
    }

    Node *nn = createNODE(data);
    Node *temp = head;

    for (int i = 1; i < pos - 1; i++) {
        if (temp == NULL) {
            printf("Position are out of range");
            return;
        }
        temp = temp->link;
    }

    nn->link = temp->link;
    temp->link = nn;
}

void deleteAtFirst() {
    if (head == NULL) {
        printf("List is empty");
        return;
    }

    Node *temp = head;
    head = head->link;
    free(temp);
}

void deleteAtEnd() {
    if (head == NULL) {
        printf("List is empty.");
        return;
    }

    if (head->link == NULL) {
        free(head);
        head = NULL;
        return;
    }

    Node *temp = head;

    while (temp->link->link != NULL)
        temp = temp->link;

    Node *last = temp->link;
```

```c
    temp->link = NULL;
    free(last);
}

void deleteAtPosition(int pos) {
    if (head == NULL) {
        printf("List is empty.");
        return;
    }

    if (pos == 1) {
        deleteAtFirst();
        return;
    }

    Node *temp = head;

    for (int i = 1; i < pos - 1; i++) {
        if (temp == NULL) {
            printf("Position out of range.");
            return;
        }
        temp = temp->link;
    }

    if (temp->link == NULL) {
        printf("Position out of range.");
        return;
    }

    Node *del = temp->link;
    temp->link = del->link;
    free(del);
}

void display() {
    if (head == NULL) {
        printf("List is empty");
        return;
    }

    Node *temp = head;

    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->link;
    }
    printf("NULL");
}

int main() {
    FILE *fp = fopen("input.txt", "r");

    if (fp == NULL) {
        printf(" file cant open please check :)");
        return 0;
    }

    int choice, data, pos;

    while (fscanf(fp, "%d", &choice) != EOF) {

        switch (choice) {
            case 1:
                fscanf(fp, "%d", &data);
                createList(data);
                break;

            case 2:
                fscanf(fp, "%d", &data);
                insertAtFirst(data);
                break;

            case 3:
                fscanf(fp, "%d", &data);
                insertAtEnd(data);
                break;

            case 4:
                fscanf(fp, "%d %d", &data, &pos);
                insertAtPosition(data, pos);
                break;

            case 5:
                deleteAtFirst();
                break;

            case 6:
                deleteAtEnd();
                break;

            case 7:
                fscanf(fp, "%d", &pos);
                deleteAtPosition(pos);
                break;
```

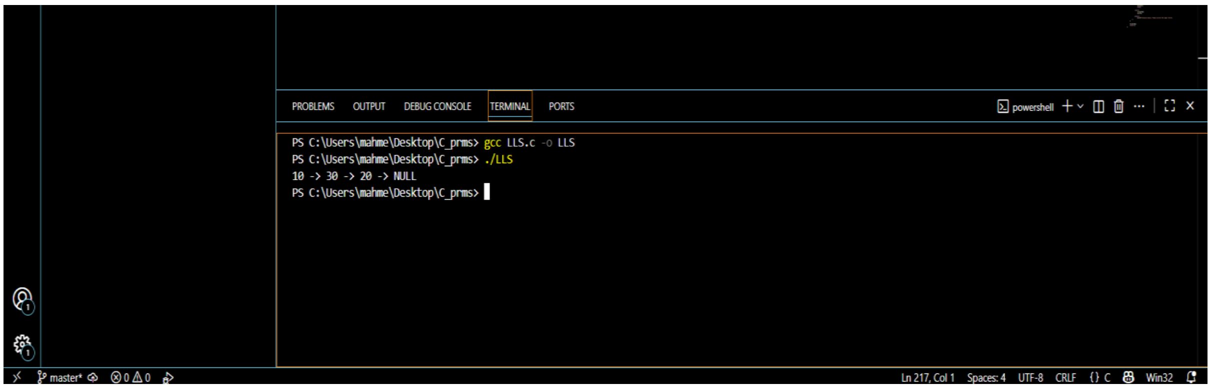
```c
        case 8:
          display();
          break;

        case 9:
          fclose(fp);
          exit(0);

        default:
          printf("Invalid choice. Please correct the input file");
      }
    }

    fclose(fp);
    return 0;
}
```

Input by file:-



```
1 10
3 20
4 30 2
8
5
9
```

OUTPUT:-



```
PS C:\Users\mahme\Desktop\C_prms> gcc LLS.c -o LLS
PS C:\Users\mahme\Desktop\C_prms> ./LLS
10 -> 30 -> 20 -> NULL
PS C:\Users\mahme\Desktop\C_prms> 
```