Problem-A:

```
#include <stdio.h>

struct list
{
    int data;     //
    list *right;  // right
    list *left;   // left
    list *top;    // top
    list *bottom; // bottom
} * Left, *Right, *Top, *Bottom;

typedef struct list node;

int main()
{
    int n, i, x = 0;
    node *head = new node;
    head->right = NULL;
    head->left = NULL;
    head->top = NULL;
    head->bottom = NULL;
    int rightSum, leftSum, topSum, bottomSum;
    printf("\nEnter the number of inputs: ");
    scanf("%d", &n);
    char c;
    int inputs = n * 2;
    printf("Enter the inputs: \n");
    for (i = 0; i < inputs; i++)
    {
        scanf("%c", &c);
        scanf("%d", &x);
        if (c == 'L')
        {

            if (head->left == NULL)
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->left = NULL;
                Left = newNode;
            }
            else
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->left = NULL;
                Left->left = newNode;
                Left = newNode;
            }

            leftSum = leftSum + x;
        }
        else if (c == 'R')
        {
            if (head->right == NULL)
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->right = NULL;
                Right = newNode;
            }
            else
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->right = NULL;
                Right->right = newNode;
                Right = newNode;
            }
            rightSum = rightSum + x;
        }
        else if (c == 'T')
        {
            if (head->top == NULL)
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->top = NULL;
                Top = newNode;
            }
            else
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->top = NULL;
                Top->top = newNode;
                Top = newNode;
            }
            topSum = topSum + x;
        }
        else
        {
            if (head->bottom == NULL)
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->bottom = NULL;
                Bottom = newNode;
            }
            else
            {
                node *newNode = new node;
                newNode->data = x;
                newNode->bottom = NULL;
                Bottom->bottom = newNode;
                Bottom = newNode;
            }
            bottomSum = bottomSum + x;
        }
    }
    int array[4] = {rightSum, leftSum, topSum, bottomSum}; // To find the
maximum value
    int max = array[0];
    for (i = 0; i < 4; i++)
    {
        if (array[i] > array[0])
        {
            max = array[i];
        }
    }
    if (max == rightSum)
    {
        printf("Right Link List Has Maximum Sum %d", rightSum);
    }
    else if (max == leftSum)
    {
        printf("Left Link List Has Maximum Sum %d", leftSum);
    }
    else if (max == topSum)
    {
        printf("Top Link List Has Maximum Sum %d", topSum);
    }
    else
```

```
        printf("Bottom Link List Has Maximum Sum %d", bottomSum);
    return 0;
}
```

Problem-B:

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

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

struct node *head = NULL, *temp, *last, *temp1;

void insert(int item)
{

    temp1 = (struct node *)malloc(sizeof(struct node));
    if (head->next == head)
    {
        last = (struct node *)malloc(sizeof(struct node));
        last->data = item;
        last->next = head;
        head->prev = last;
        head->next = last;
        last->prev = head;
    }
    else if (item <= head->next->data)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->data = item;
        temp->next = head->next;
        temp->prev = head;
        temp->next->prev = temp;
        head->next = temp;
    }
    else
    {
        temp1 = (struct node *)malloc(sizeof(struct node));
        temp = head->next;
        while (temp->data <= item)
        {
            temp->next = temp;
        }
        temp1->data = item;
        temp1->prev = temp;
        temp1->next = temp->next;
        temp->next = temp1;
        temp1->next->prev = temp1;
    }
}

int length()
{
    int k = 1;
    if (head->next != head)
    {
        temp = head->next;
        while (temp->next != head)
        {
            k++;
            temp = temp->next;
        }
    }
    return k;
}

void deleteNode(int item)
{
    temp = head->next;
    if (head->next == head)
        printf("Doubly linked list is empty");
    else if (head->next->next == head)
        head->next = NULL;
    else
    {
        while ((temp->next != head) && (temp->data != item))
        {
            temp = temp->next;
        }
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
        free(temp);
    }
}
```

```c
void display()
{
    temp = head->next;

    while (temp->next != head)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("%d", head->prev->data);
    printf("\n");
}

void search(int item)
{
    temp = head->next;
    while ((temp->next != head) && (temp->data != item))
    {
        temp = temp->next;
    }
    if (temp->data == item)
    {
        printf("Integer found");
    }
    else
    {
        printf("Integer not found");
    }
}

void showMenu()
{
    printf("\n1. Insert\n2. Show all\n3. Search\n4. Delete\n5. Length\n6.
Exit\n");
}

int main()
{
    int choice, i = 1, item;
    head = (struct node *)malloc(sizeof(struct node));
    head->next = head;
    do
    {
        showMenu();
        printf("Enter your choice: ");
        scanf("%d", &choice);
        if (choice == 1)
        {
            printf("Insert: ");
            scanf("%d", &item);
            insert(item);
            display();
        }
        else if (choice == 2)
        {
            printf("Show all: ");

            display();
        }
        else if (choice == 3)
        {
            printf("Search: ");
            search(item);
        }
        else if (choice == 4)
        {
            printf("Delete: ");
            deleteNode(item);
        }
        else if (choice == 5)
        {
            printf("Length: ");

            printf("The length is: %d", length());
        }
        else
        {
            i = 0;
        }

    } while (i == 1);

    return 0;
}
```

Problem-C:

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

```c
struct list
{
    int data;
    struct list *next;
    struct list *prev;
} * dummy;

typedef struct list node;

int flag;

void insert(int y, int x)
{
    node *temp, *temp1 = (node *)malloc(sizeof(node));

    if (dummy->next == NULL)
    {
        temp1->data = x;
        temp1->next = dummy->next;
        temp1->prev = dummy;
        dummy->next = temp1;
        printf("Insert after dummy node\n");
    }

    else
    {
        flag = 0;
        temp = dummy->next;
        while (temp != NULL)
        {
            if (temp->data == y)
            {
                flag = 1; // used flag to find the y node
                break;
            }
            temp = temp->next;
        }
        if (flag == 0)
        {
            node *newNode = (node *)malloc(sizeof(node)); // Allocated new
memory as the temp1 is used before
            newNode->data = x;
            newNode->prev = dummy;
            newNode->next = dummy->next;
            dummy->next = newNode;
            printf("Insert after dummy node\n");
        }
        else
        {
            node *newNode = (node *)malloc(sizeof(node)); // Allocated new
memory as the temp1 and newNode is used before
            newNode->data = x;
            newNode->prev = temp;
            newNode->next = temp->next;
            if (temp->next != NULL)
            {
                temp->next->prev = newNode;
            }

            temp->next = newNode;
            printf("Insert after %d\n", y);
        }
    }
}

void deleter(int x)
{
    node *temp, *prev;
    if (dummy->next == NULL)
    {
        return;
    }
    else
    {
        temp = dummy->next;
        prev = dummy;
        while ((temp != NULL) && (temp->data != x))
        {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL)
        {
            printf("Delete not possible\n");
        }
        else
        {
            prev->next = temp->next;
            if (temp->next != NULL)
            {
                prev->next->prev = prev;
            }

            printf("Node with key value %d is DELETED\n", temp->data);
            free(temp);
        }
    }
}

void search(int item)
{
    node *temp;
    temp = dummy->next;
    while ((temp != NULL) && (temp->data != item))
    {
        temp = temp->next;
    }
    if (temp == NULL)
    {
        printf("Not FOUND\n");
    }
    else
    {
        printf("Node with key value %d is FOUNDED", temp->data);
    }
}

void show()
{
    node *temp;
    temp = dummy->next;
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

void showMenu()
{
    printf("\n1. insert\n2. remove\n3. search\n4. showall\n5. exit\n");
}

int main()
```

```c
{
    int choice, i = 1, item;
    int x, y;
    do
    {
        showMenu();
        printf("Enter your choice: ");
        scanf("%d", &choice);
        if (choice == 1)
        {
            printf("ins : ");
            scanf("%d", &y);
            scanf("%d", &x);
            insert(y, x);
        }
        else if (choice == 2)
        {
            printf("del : ");
            scanf("%d", &x);
            deleter(x);
        }
        else if (choice == 3)
        {
            printf("sch : ");
            scanf("%d", &item);
            search(item);
        }
        else if (choice == 4)
        {
            printf("shw ");
            show();
        }

        else
        {
            i = 0;
        }

    } while (i == 1);

    return 0;
}
```

Problem-D:

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

struct node
{
    char character;
    int row;
    int counter;
    struct node *next;
    struct node *down;
} * dummy;

typedef struct node node;

void mirror(int row, int column)
{
    node *head = dummy, *tail, *newNode, *temp, *temp1;
    char character[100];
    for (int i = 0; i < row; i++)
    {
        scanf("%s", character);
        for (int j = 0; j < column; j++)
        {

            if (head->next == NULL)
            {
                newNode = (node *)malloc(sizeof(node));
                newNode->character = character[j];
                newNode->row = j;
                newNode->next = NULL;
                head->next = newNode;
            }
            else
            {
                newNode = (node *)malloc(sizeof(node));
                newNode->character = character[j];
                newNode->row = j;
                newNode->next = tail; // adding this new node before tail
                head->next = newNode; // adding right after the head node
to print in reverse order
                tail = newNode;       // assigning the new node as tail
for inserting the next new node between head
                and tail
            }
        }
        newNode = (node *)malloc(sizeof(node));
        head->down = newNode;
        head = newNode;
        head->next = NULL;
    }
    printf("\n");
    // for printing the list
    temp = dummy;
    for (int i = 0; i < row; i++)
    {
        temp1 = temp->next;
        for (int j = 0; j < column; j++)
        {
            printf("%c", temp->character);
            temp1 = temp1->next;
        }
        printf("\n");
        temp = temp->down;
    }
}

int main()
{
    int row, column;
    dummy = (node *)malloc(sizeof(node));
    dummy->next = NULL;
    scanf("%d %d", &row, &column);
    mirror(row, column);
    return 0;
}
```

Problem-E:

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

struct node
{
    char character;
    int count;
    struct node *next;
```

```c
} * dummy;

typedef struct node node;

// Creates the list of alphabets
void createList()
{
    node *temp, *newNode;
    char character;
    int i = 0;
    for (i = 0; i < 26; i++)
    {
        if (dummy->next == NULL)
        {
            newNode = (node *)malloc(sizeof(node));
            newNode->character = 'a' + i;
            newNode->count = 0;
            newNode->next = NULL;
            dummy->next = newNode;
            temp = newNode;
        }
        else
        {
            newNode = (node *)malloc(sizeof(node));
            newNode->character = 'a' + i;
            newNode->count = 0;
            newNode->next = NULL;
            temp->next = newNode;
            temp = newNode;
        }
    }
}

// search alphabets, if found increment the count value
void searchList(char chararray[])
{

    node *temp;
    temp = dummy->next;
    int i = 0;
    while (chararray[i])
    {
        if ('a' <= chararray[i] && chararray[i] <= 'z')
        {
            char character = chararray[i];
            while (temp != NULL)
            {
                if (temp->character == character)
                {
                    temp->count++;
                    break;
                }
                temp = temp->next;
            }
            i++;
        }
    }
}

void displayList()
{
    node *temp;
    temp = dummy->next;
    while (temp != NULL)
    {
        printf("%c : %d\n", temp->character, temp->count);
        temp = temp->next;
    }
}

int main()
{
    char chararray[100];
    dummy = (node *)malloc(sizeof(node));
    dummy->next = NULL;
    scanf("%s", chararray);
    createList();
    searchList(chararray);
    displayList();
    return 0;
}
```

Problem-F:

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

struct node
```

```
{
    int coefficient;
    int power;
    struct node *next;
} typedef node;

void addNode(node **head, int x, int y)
{
    node *temp = (node *)malloc(sizeof(node));
    temp->coefficient = x;
    temp->power = y;
    temp->next = NULL;
    if (*head == NULL)
        *head = temp;
    else
    {
        node *temp1 = *head;
        while (temp1->next != NULL)
        {
            temp1 = temp1->next;
        }
        temp1->next = temp;
    }
}

void printList(node *head)
{
    node *temp = head;
    printf("head -> ");
    while (temp != NULL)
    {
        printf("[%d %d] ", temp->coefficient, temp->power);
        temp = temp->next;
    }
    printf(" <- tail\n");
}

node *addPolynomials(node *eqn1, node *eqn2)
{
    node *head = NULL;
    node *temp = NULL;
    while (eqn1 != NULL && eqn2 != NULL)
    {
        if (eqn1->power == eqn2->power)
        {
            addNode(&head, eqn1->coefficient + eqn2->coefficient, eqn1-
>power);
            eqn1 = eqn1->next;
            eqn2 = eqn2->next;
        }
        else if (eqn1->power > eqn2->power)
        {
            addNode(&head, eqn1->coefficient, eqn1->power);
            eqn1 = eqn1->next;
        }
        else
        {
            addNode(&head, eqn2->coefficient, eqn2->power);
            eqn2 = eqn2->next;
        }
    }
    while (eqn1 != NULL)
    {
        addNode(&head, eqn1->coefficient, eqn1->power);
        eqn1 = eqn1->next;
    }
    while (eqn2 != NULL)
    {
        addNode(&head, eqn2->coefficient, eqn2->power);
        eqn2 = eqn2->next;
    }
    return head;
}

int main()
{
    node *eqn1 = NULL;
    node *eqn2 = NULL;
    addNode(&eqn1, 5, 3);
    addNode(&eqn1, 2, 2);
    addNode(&eqn1, 3, 1);
    addNode(&eqn1, 4, 0);
    addNode(&eqn2, 4, 2);
    addNode(&eqn2, 3, 0);
    printList(eqn1);
    printList(eqn2);
    node *eqn3 = addPolynomials(eqn1, eqn2);
    printList(eqn3);
}
```