Q1. Write a program to implement all the basic operations of a stack using linked list

Program Code:

```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node{
    int val;
    struct node *next;
};
struct node *head;
void main(){
    int choice = 0;
    printf("\n----Stack Menu----\n");
    while (choice != 4){
        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice){
        case 1:{
            push();
            break;
        }
        case 2:{
            pop();
            break;
        }
        case 3:{
            display();
            break;
        }
        case 4:{
            printf("Exiting....");
            break;
        }
        default:{
            printf("Please Enter valid choice ");
        }
        };
    }
}
void push(){
    int val;
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL){
        printf("Not able to push the element.");
    }
    else{
        printf("Enter the value: ");
        scanf("%d", &val);
        if (head == NULL){
            ptr->val = val;
            ptr->next = NULL;
            head = ptr;
        }
        else{
            ptr->val = val;
            ptr->next = head;
            head = ptr;
        }
        printf("Item pushed.");
    }
}
void pop(){
    int item;
    struct node *ptr;
    if (head == NULL){
        printf("Underflow");
    }
    else{
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped.");
    }
}
void display(){
    int i;
    struct node *ptr;
    ptr = head;
    if (ptr == NULL){
        printf("Stack is empty.\n");
    }
    else{
        printf("Printing Stack elements...\n");
        while (ptr != NULL){
            printf("%d\n", ptr->val);
            ptr = ptr->next;
  }
} }
```

Q2. Write a program to implement tall the operations of a queue using linked list

Program Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int info;
    struct node *ptr;
} * front, *rear, *temp, *front1;
int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();
int count = 0;
void main(){
    int no, ch, e;
    printf("\n----Queue Menu----\n");
    printf("\n1.Enque\n2.Deque\n3.Front
element\n4.Empty\n5.Exit\n6.Display\n7.Queue size");
    create();
    while (1){
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch (ch){
        case 1:
            printf("\nEnter element: ");
            scanf("%d", &no);
            enq(no);
            break;
        case 2:
            deq();
            break;
        case 3:
            e = frontelement();
            if (e != 0)
                printf("Front element: %d", e);
            else
                printf("\nNo front element in Queue as queue is
empty.");
            break;
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            queuesize();
            break;
        default:
            printf("Wrong choice, please enter correct choice.");
            break;
        }
    }
}
void create(){
    front = rear = NULL;
}
void queuesize(){
    printf("\n Queue size: %d", count);
}
void enq(int data){
    if (rear == NULL){
        rear = (struct node *)malloc(1 * sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }
    else{
        temp = (struct node *)malloc(1 * sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
        temp->ptr = NULL;
        rear = temp;
    }
    count++;
}
void display(){
    front1 = front;

    if ((front1 == NULL) && (rear == NULL)){
        printf("Queue is empty");
        return;
    }
    while (front1 != rear){
        printf("%d ", front1->info);
        front1 = front1->ptr;
    }
    if (front1 == rear)
```

```
        printf("%d", front1->info);
}
void deq(){
    front1 = front;

    if (front1 == NULL){
        printf("\n Error! queue is empty");
        return;
    }
    else if (front1->ptr != NULL){
        front1 = front1->ptr;
        printf("\n Dequed value: %d", front->info);
        free(front);
        front = front1;
    }
    else{
        printf("\n Dequed value : %d", front->info);
        free(front);
        front = NULL;
        rear = NULL;
    }
    count--;
}
int frontelement(){
    if ((front != NULL) && (rear != NULL))
        return (front->info);
    else
        return 0;
}
void empty(){
    if ((front == NULL) && (rear == NULL))
        printf("\n Queue is empty.");
    else
        printf("Queue is not empty.");
}
```

Q3. Write a program to add two polynomials using linked list.

Program Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node{
    int coeff;
    int pow;
    struct Node *next;
};
void readPolynomial(struct Node **poly){
    int coeff, exp, cont;
    struct Node *temp = (struct Node *)malloc(sizeof(struct Node));
    *poly = temp;
    do{
        printf("\n Coeffecient: ");
        scanf("%d", &coeff);
        printf("\n Exponent: ");
        scanf("%d", &exp);
        temp->coeff = coeff;
        temp->pow = exp;
        temp->next = NULL;
        printf("\nHave more terms?\nPress 1 for YES and 0 for NO: ");
        scanf("%d", &cont);
        if (cont){
            temp->next = (struct Node *)malloc(sizeof(struct Node));
            temp = temp->next;
            temp->next = NULL;
        }
    } while (cont);
}
void displayPolynomial(struct Node *poly){
    printf("\nPolynomial expression: ");
    while (poly != NULL){
        printf("%dX^%d", poly->coeff, poly->pow);
        poly = poly->next;
        if (poly != NULL)
            printf("+");
    }
}
void addPolynomials(struct Node **result, struct Node *first, struct
Node *second){
    struct Node *temp = (struct Node *)malloc(sizeof(struct Node));
    temp->next = NULL;
    *result = temp;
    while (first && second){
        if (first->pow > second->pow){
            temp->coeff = first->coeff;
            temp->pow = first->pow;
            first = first->next;
        }
        else if (first->pow < second->pow){
            temp->coeff = second->coeff;
            temp->pow = second->pow;
            second = second->next;
        }
        else{
            temp->coeff = first->coeff + second->coeff;
            temp->pow = first->pow;
            first = first->next;
            second = second->next;
        }
        if (first && second){
            temp->next = (struct Node *)malloc(sizeof(struct Node));
            temp = temp->next;
            temp->next = NULL;
        }
    }
    while (first || second){
        temp->next = (struct Node *)malloc(sizeof(struct Node));
        temp = temp->next;
        temp->next = NULL;

        if (first){
            temp->coeff = first->coeff;
            temp->pow = first->pow;
            first = first->next;
        }

        else if (second){
            temp->coeff = second->coeff;
            temp->pow = second->pow;
            second = second->next;
        }
    }
}
int main(){
    struct Node *first = NULL;
    struct Node *second = NULL;
    struct Node *result = NULL;
    printf("\nFirst polynomial: \n");
    readPolynomial(&first);
    displayPolynomial(first);
    printf("\nSecond polynomial: \n");
    readPolynomial(&second);
    displayPolynomial(second);
    addPolynomials(&result, first, second);
    displayPolynomial(result);
    return 0;
}
```

Q4. Write a program to multiply two polynomials using linked list.

Program Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int coefficient, exponent;
    struct node *next;
};
struct node *hPtr1, *hPtr2, *hPtr3;
struct node *buildNode(int coefficient, int exponent){
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    ptr->coefficient = coefficient;
    ptr->exponent = exponent;
    ptr->next = NULL;
    return ptr;
}
void polynomial_insert(struct node **myNode, int coefficient, int
exponent){
    struct node *lPtr, *pPtr, *qPtr = *myNode;
    lPtr = buildNode(coefficient, exponent);
    if (*myNode == NULL || (*myNode)->exponent < exponent){
        *myNode = lPtr;
        (*myNode)->next = qPtr;
        return;
    }
    while (qPtr){
        pPtr = qPtr;
        qPtr = qPtr->next;
        if (!qPtr){
            pPtr->next = lPtr;
            break;
        }
        else if ((exponent < pPtr->exponent) && (exponent >
qPtr->exponent)){
            lPtr->next = qPtr;
            pPtr->next = lPtr;
            break;
        }
    }
    return;
}
void polynomial_add(struct node **n1, int coefficient, int exponent){
    struct node *x = NULL, *temp = *n1;
    if (*n1 == NULL || (*n1)->exponent < exponent){
        *n1 = x = buildNode(coefficient, exponent);
        (*n1)->next = temp;
    }
    else{
        while (temp){
            if (temp->exponent == exponent){
                temp->coefficient = temp->coefficient + coefficient;
                return;
            }
            if (temp->exponent > exponent && (!temp->next ||
temp->next->exponent < exponent)){
                x = buildNode(coefficient, exponent);
                x->next = temp->next;
                temp->next = x;
                return;
            }
            temp = temp->next;
        }
        x->next = NULL;
        temp->next = x;
    }
}
void polynomial_multiply(struct node **n1, struct node *n2, struct
node *n3){
    struct node *temp;
    int coefficient, exponent;
    temp = n3;
    if (!n2 && !n3){
        return;
    }
    if (!n2){
        *n1 = n3;
    }
    else if (!n3){
        *n1 = n2;
    }
    else{
        while (n2){
            while (n3){
                coefficient = n2->coefficient * n3->coefficient;
                exponent = n2->exponent + n3->exponent;
                n3 = n3->next;
                polynomial_add(n1, coefficient, exponent);
            }
            n3 = temp;
            n2 = n2->next;
        }
    }
    return;
}
struct node *polynomial_deleteList(struct node *ptr){
    struct node *temp;
    while (ptr){
        temp = ptr->next;
        free(ptr);
        ptr = temp;
    }
    return NULL;
}
void polynomial_view(struct node *ptr){
    int i = 0;
    int flag = 0;
    while (ptr){
        if (ptr->exponent != 0 || ptr->exponent != 1){
            if (ptr->coefficient > 0 && flag == 0){
                printf("%dx^%d", ptr->coefficient, ptr->exponent);
                flag++;
            }
            else if (ptr->coefficient > 0 && flag == 1){
                printf("+%dx^%d", ptr->coefficient, ptr->exponent);
            }
            else if (ptr->coefficient < 0){
                printf("%dx^%d", ptr->coefficient, ptr->exponent);
            }
        }
        else if (ptr->exponent == 0){
            if (ptr->coefficient > 0 && flag == 0){
                printf("%d", ptr->coefficient);
                flag++;
            }
            else if (ptr->coefficient > 0 && flag == 1){
                printf("+%d", ptr->coefficient);
            }
            else if (ptr->coefficient < 0){
                printf("%d", ptr->coefficient);
            }
        }
        else if (ptr->exponent == 1){
            if (ptr->coefficient > 0 && flag == 0){
                printf("%dx", ptr->coefficient);
                flag++;
            }
            else if (ptr->coefficient > 0 && flag == 1){
                printf("+%dx", ptr->coefficient);
            }
            else if (ptr->coefficient < 0){
                printf("%dx", ptr->coefficient);
            }
        }
        ptr = ptr->next;
        i++;
    }
    printf("\n");
    return;
}
int main(int argc, char *argv[]){
    int coefficient, exponent, i, n;
    int count;
    printf("Enter the no. of coefficients in the multiplicand:");
    scanf("%d", &count);
    for (i = 0; i < count; i++){
        printf("Enter coefficient: ");
        scanf("%d", &coefficient);
        printf("Enter exponent: ");
        scanf("%d", &exponent);
        polynomial_insert(&hPtr1, coefficient, exponent);
    }
    printf("Enter the no. of coefficients in the multiplier:");
    scanf("%d", &count);
    for (i = 0; i < count; i++){
        printf("Enter coefficient: ");
        scanf("%d", &coefficient);
        printf("Enter exponent: ");
        scanf("%d", &exponent);
        polynomial_insert(&hPtr2, coefficient, exponent);
    }
    printf("First Polynomial Expression: ");
    polynomial_view(hPtr1);
    printf("Second Polynomial Expression: ");
    polynomial_view(hPtr2);
    polynomial_multiply(&hPtr3, hPtr1, hPtr2);
    printf("Output:\n");
    polynomial_view(hPtr3);
    hPtr1 = polynomial_deleteList(hPtr1);
    hPtr2 = polynomial_deleteList(hPtr2);
    hPtr3 = polynomial_deleteList(hPtr3);
    return 0;
}
```