**ASSIGNMENET 01**

**CSE THEORY**

**Kumar Ayush**

**21051737**

**CSE 18**

**Q1. Sort the array of elements in ascending order based on their frequency?**

**Ex- input: 2, 4, 2, 8, 9, 8, 8, 4, 4, 8**

**Output: 9, 2, 4, 8**

#include <stdio.h>

#include<stdlib.h>

struct node{

    int no;

    int freq;

    struct node\* next;

};

struct node\* head = NULL;

int main(){

    int n;

    printf("\nenter the the size of size : ");

    scanf("%d",&n);

    int a[n] , f[n];

    for(int i=0 ; i<n ; i++){

        scanf("%d",&a[i]);

    }

    f[0] = a[0];

    int c = 1 , b;

    for(int i=0 ; i<n ; i++){

        b=0;

        for(int j=0 ; j<c ; j++){

            if(a[i] == f[j]){

                b++;

                break;

            }

        }

        if(b == 0)

        f[c++] = a[i];

    }

    int sort[c];

    for(int j=0 ; j<c ; j++){

        int n\_f = 0;

        for(int i=0 ; i<n ; i++){

            if(f[j] == a[i])

            n\_f++;

        }

        sort[j] = n\_f;

    }

    struct node \*newnnode , \*ptr , \*temp , \*prev;

    for(int i=0 ; i<c ; i++){

        newnnode = (struct node\*)malloc(sizeof(struct node));

        newnnode->no = f[i];

        newnnode->freq = sort[i];

        newnnode->next = NULL;

        if(head == NULL)

        head = ptr = newnnode;

        else

        ptr = ptr->next = newnnode;

    }

    for(int i=0 ; i<c ; i++){

        ptr = head;

        for(int j=0 ; j<c-1 ; j++){

            if(ptr->freq > ptr->next->freq){

                int temp1 = ptr->no ;

                int temp2 = ptr->freq;

                ptr->freq = ptr->next->freq;

                ptr->no = ptr->next->no;

                ptr->next->freq = temp2;

                ptr->next->no = temp1;

            }

            ptr = ptr->next;

        }

    }

    ptr = head;

    for(int i=0 ; i<c ; i++){

        printf("%d ",ptr->no);

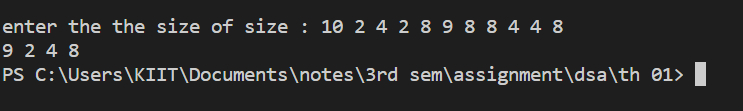
        ptr = ptr->next;

    }

    return 0;

}

**OUTPUT**



**Q2. Write a program to check an expression if it is balanced parenthesize or not?**

#include <stdio.h>

#include <stdlib.h>

void push(int c);

void pop();

void dis\_stack();

struct node{

    char para;

    struct node \*next;

};

struct node\* top = NULL;

int n\_node = 0;

void push(char c){

   // printf("\nb: %c",c);

    struct node\* newnode;

    n\_node++;

    newnode = (struct node\*)malloc(sizeof(struct node));

    newnode->para = c;

    newnode->next = NULL;

    if(top == NULL)

        top = newnode;

    else{

        newnode->next = top;

        top = newnode;

    }

    //dis\_stack();

}

void dis\_stack(){

    struct node \*ptr = top;

    while(ptr != NULL){

        printf(" %d",ptr->para);

        ptr = ptr->next;

    }

    printf("\n");

}

void pop(){

    struct node \*ptr = top;

    top = top->next;

    free(ptr);

    n\_node--;

    /\*printf("\ns:");

    dis\_stack();\*/

}

int main(){

    char c[100];

    int last = 0 , p = 0;

    printf("enter your expression : ");

    scanf("%s",&c);

    int i=0 ;

    while(c[i] != '\0'){

        if((c[i] != ')') && (c[i] != ']') && (c[i] != '}')){

            char ch = c[i];

            push(ch);

            if((c[i] == '(') || (c[i] == '[') || (c[i] == '{'))

            p++;

        }

        else{

            if(p == 0 ){

                last++;

            }

            else{

                while((top->para != '(') && (top->para != '[') && (top->para != '{')){

                    pop();

                }

                if((c[i] == ')') && (top->para == '(')){

                    pop();

                    p--;

                }

                else if((c[i] == '}') && (top->para == '{')){

                    pop();

                    p--;

                }

                else if((c[i] == ']') && (top->para == '[')){

                    pop();

                    p--;

                }

                else

                last++;

            }

        }

        if(last == 0)

            i++;

        else{

            printf("\nthe expression is not balanced parenthesize");

            break;

        }

    }

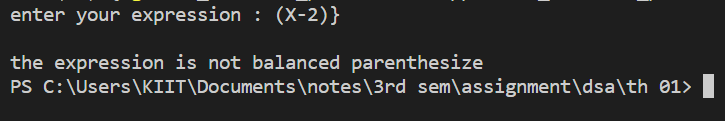
    if(last == 0)

    printf("the expression is balanced parethesize");

    return 0;

}

**OUTPUT**



**Q3. Write a menu driven program to find the union and intersection of two linked**

**list.**

#include <stdio.h>

#include <stdlib.h>

struct node{

    int data;

    struct node\* next;

};

void create\_set(struct node \*head , int n\_node){

    struct node \*ptr , \*newnode;

    ptr = head;

    for(int i=0 ; i<n\_node ; i++){

        newnode = (struct node\*)malloc(sizeof(struct node));

        scanf("%d",&newnode->data);

        newnode->next = NULL;

        if(head == NULL)

        head = ptr = newnode;

        else

        ptr = ptr->next = newnode;

    }

}

void dis\_set(struct node \*head){

    struct node \*ptr = head->next;

    while(ptr != NULL){

        printf(" %d",ptr->data);

        ptr = ptr->next;

    }

}

void set\_op(struct node \*head1 , struct node \*head2 , int n1 ,int n2){

    struct node \*ptr1 , \*ptr2;

    int c , n\_i = 0 , n\_u = 0;

    int in[n1+n2] , u[n1+n2];

    ptr1 = head1->next;

    for(int i=0 ; i<n1 ; i++ ){

        ptr2 = head2->next;

        c = 0;

        u[n\_u++] = ptr1->data;

        for(int j=0 ; j<n2 ; j++){

            if(ptr1->data == ptr2->data)

            c++;

            ptr2 = ptr2->next;

        }

        if(c > 0)

        in[n\_i++] = ptr1->data;

        ptr1 = ptr1->next;

    }

    ptr2 = head2->next;

    while(ptr2 !=NULL){

        c=0;

        for(int j=0 ; j<n\_u ; j++){

            if(ptr2->data == u[j])

            c++;

        }

        if(c == 0)

        u[n\_u++] = ptr2->data;

        ptr2 = ptr2->next;

    }

    printf("\nunion of set 1 and set 2 is : ");

    for(int i=0 ; i<n\_u ; i++)

    printf(" %d",u[i]);

    printf("\nintersection of set 1 and set 2 is : ");

    for(int i=0 ; i<n\_i ; i++)

    printf(" %d",in[i]);

}

int main(){

    struct node \*head1 = NULL , \*head2 = NULL;

    int n\_node1 = 0 , n\_node2 = 0;

    printf("\nenter no. elements for set 1 and se 2 respectively : ");

    scanf("%d%d",&n\_node1 , &n\_node2);

    printf("\n---------input sets------------\n");

    printf("\nenter set 1 : ");

    create\_set(&head1 , n\_node1);

    printf("\nenter set 2 : ");

    create\_set(&head2 , n\_node2);

    printf("\nset 1 : ");

    dis\_set(&head1);

    printf("\nset 2 : ");

    dis\_set(&head2);

    printf("\n----------set operation------------\n");

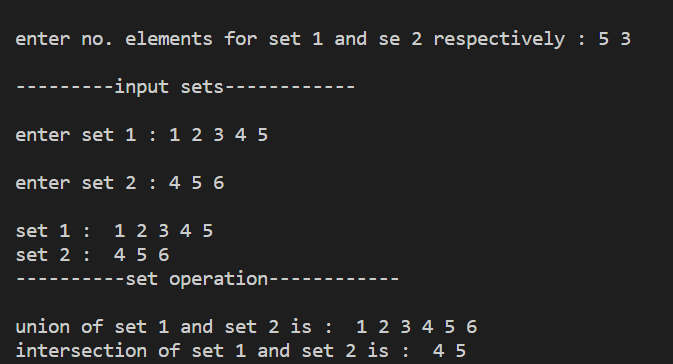
    set\_op(&head1 , &head2 , n\_node1 , n\_node2);

    return 0;

}

//5 3 1 2 3 4 5 4 5 6

**OUTPUT**



**Q4. Write a program to check whether a number represented by linked list is a**

**palindrome or not.**

#include <stdio.h>

#include <stdlib.h>

struct node{

    int data;

    struct node\* next;

    struct node \*prev;

};

struct node \*head = NULL;

struct node\* tail = NULL;

int n\_node = 0;

void create\_list(int n){

    struct node \*newnode;

    while(n > 0){

        newnode = (struct node\*)malloc(sizeof(struct node));

        newnode->data = n%10;

        newnode->next = newnode->prev = NULL;

        if(head == NULL)

        tail = head = newnode;

        else{

            newnode->prev = tail;

            tail = tail->next = newnode;

        }

        n\_node++;

        n /= 10;

    }

}

void dis\_list(){

    struct node \*ptr = tail;

    while(ptr != NULL){

        printf(" %d",ptr->data);

        ptr = ptr->prev;

    }

}

void check\_palin(){

    struct node \*ptr1 = head , \*ptr2 = tail;

    int c = 0;

    for(int i=0 ; i < n\_node/2 ; i++){

        if(ptr1->data != ptr2->data){

            c++;

            break;

        }

    }

    if(c == 0 ){

        printf("\nthe no. is palindrome");

    }

    else

    printf("\nthe no. is not palindrome");

}

int main(){

    int n;

    printf("enter the no. : ");

    scanf("%d",&n);

    create\_list(n);

    printf("the no. is : ");

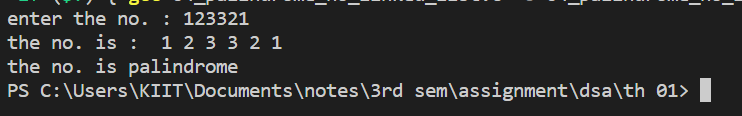
    dis\_list();

    check\_palin();

    return 0;

}

**OUTPUT**



**Q5. Write a program (recursive and non-recursive) to convert infix to prefix as well**

**as convert the following infix to equivalent prefix expression using stack.**

**9\*(2+1)^2/(5-8/2/2)**

#include<stdio.h>

#include<string.h>

#include<limits.h>

#include<stdlib.h>

# define MAX 100

int top = -1;

char stack[MAX];

// checking if stack is full

int isFull() {

    return top == MAX - 1;

}

// checking is stack is empty

int isEmpty() {

    return top == -1;

}

// Push function here, inserts value in stack and increments stack top by 1

void push(char item) {

    if (isFull())

        return;

    top++;

    stack[top] = item;

}

// Function to remove an item from stack.  It decreases top by 1

int pop() {

    if (isEmpty())

        return INT\_MIN;

    // decrements top and returns what has been popped

    return stack[top--];

}

// Function to return the top from stack without removing it

int peek(){

    if (isEmpty())

        return INT\_MIN;

    return stack[top];

}

// A utility function to check if the given character is operand

int checkIfOperand(char ch) {

    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');

}

// Fucntion to compare precedence

// If we return larger value means higher precedence

int precedence(char ch)

{

    switch (ch)

    {

    case '+':

    case '-':

        return 1;

    case '\*':

    case '/':

        return 2;

    case '^':

        return 3;

    }

    return -1;

}

// The driver function for infix to postfix conversion

int getPostfix(char\* expression)

{

    int i, j;

    for (i = 0, j = -1; expression[i]; ++i)

    {

        // Here we are checking is the character we scanned is operand or not

        // and this adding to to output.

        if (checkIfOperand(expression[i]))

            expression[++j] = expression[i];

        // Here, if we scan character ‘(‘, we need push it to the stack.

        else if (expression[i] == '(')

            push(expression[i]);

        // Here, if we scan character is an ‘)’, we need to pop and print from the stack

        // do this until an ‘(‘ is encountered in the stack.

        else if (expression[i] == ')')

        {

            while (!isEmpty(stack) && peek(stack) != '(')

                expression[++j] = pop(stack);

            if (!isEmpty(stack) && peek(stack) != '(')

                return -1; // invalid expression

            else

                pop(stack);

        }

        else // if an opertor

        {

            while (!isEmpty(stack) && precedence(expression[i]) <= precedence(peek(stack)))

                expression[++j] = pop(stack);

            push(expression[i]);

        }

    }

    // Once all inital expression characters are traversed

    // adding all left elements from stack to exp

    while (!isEmpty(stack))

        expression[++j] = pop(stack);

    expression[++j] = '\0';

}

void reverse(char \*exp){

    int size = strlen(exp);

    int j = size, i=0;

    char temp[size];

    temp[j--]='\0';

    while(exp[i]!='\0')

    {

        temp[j] = exp[i];

        j--;

        i++;

    }

    strcpy(exp,temp);

}

void brackets(char\* exp){

    int i = 0;

    while(exp[i]!='\0')

    {

        if(exp[i]=='(')

            exp[i]=')';

        else if(exp[i]==')')

            exp[i]='(';

        i++;

    }

}

void InfixtoPrefix(char \*exp){

    int size = strlen(exp);

    // reverse string

    reverse(exp);

    //change brackets

    brackets(exp);

    //get postfix

    getPostfix(exp);

    // reverse string again

    reverse(exp);

}

int main()

{

    printf("The infix is: ");

    char expression[] = "(9\*(2+1)^2/(5-8/2/2))";

    printf("%s\n",expression);

    InfixtoPrefix(expression);

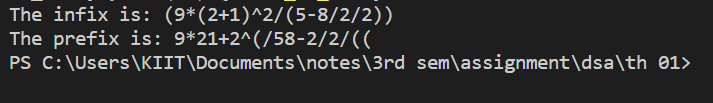
    printf("The prefix is: ");

    printf("%s\n",expression);

    return 0;

}

**OUTPUT**



**Q6. Write a program to implement queue using stacks.**

#include <stdio.h>

#include <stdlib.h>

struct stack1{

    int data;

    struct stack1 \*next;

};

struct stack2{

    int data;

    struct stack2 \*next;

};

struct stack1 \*f1 = NULL;

struct stack2 \*f2 = NULL;

int n = 0;

void push(int x){

    struct stack1 \*newnode = (struct stack1\*)malloc(sizeof(struct stack1));

    newnode->data = x;

    newnode->next = NULL;

    if(f1 == NULL)

    f1 = newnode;

    else

    {

        newnode->next = f1;

        f1 = newnode;

    }

    n++;

}

void pop(){

    for(int i=0 ; i<n ; i++){

        struct stack1 \*ptr = f1;

        struct stack2 \*newnode = (struct stack2\*)malloc(sizeof(struct stack2));

        newnode->data = f1->data;

        newnode->next = NULL;

        if(f2 == NULL)

        f2 = newnode;

        else{

            newnode->next = f2;

            f2 = newnode;

        }

        f1 = f1->next;

        free(ptr);

    }

    printf("\n\npoping out %d from the stack",f2->data);

    struct stack2 \*ptr = f2;

    f2 = f2->next;

    free(ptr);

    f1 = NULL;

    n--;

    int k = n;

    n = 0;

    for(int i=0 ; i<k ; i++){

        push(f2->data);

        struct stack2 \*ptr = f2;

        f2 = f2->next;

        free(ptr);

    }

    f2 = NULL;

}

void dis(){

    if(f1 == NULL){

        printf("\nthe list is empty");

        return;

    }

    else{

        struct stack1 \*ptr = f1;

        printf("\nthe stack is : ");

        while(ptr != NULL){

            printf("%d ",ptr->data);

            ptr = ptr->next;

        }

    }

}

int main(){

    char ch;

    int c , x;

    do{

        printf("\n\n1 push\n2 pop\n3 display\nenter choice : ");

        scanf("%d",&c);

        switch(c){

            case 1 :

                printf("enter value to push : ");

                scanf("%d",&x);

                push(x);

                break;

            case 2 :

                pop();

                break;

            case 3 :

                dis();

                break;

            default :

                printf("\nenter a valid option");

                break;

        }

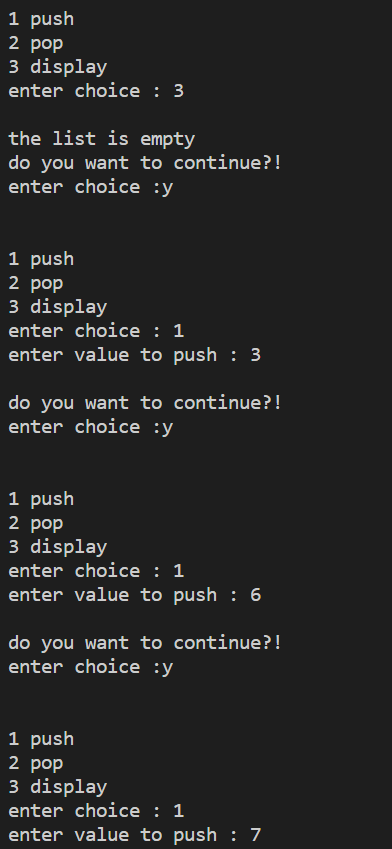
        printf("\ndo you want to continue?!\nenter choice :");

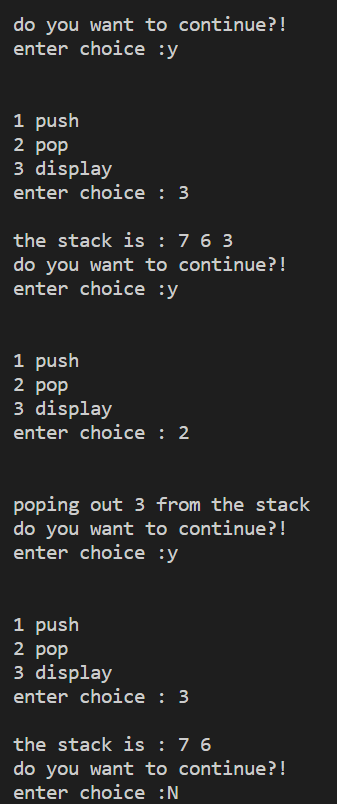
        scanf(" %c",&ch);

    }while((ch=='y')||(ch=='Y'));

}

**OUTPUT**





**Q7. Write a program to perform multiplication of two polynomials.**

#include <stdio.h>

int main(){

    int n1 , n2 , m=0;

    printf("enter size of 1st polynomial : ");

    scanf("%d",&n1);

    int a[n1];

    printf("enter the 1st polynomial : ");

    for(int i=n1-1 ; i>=0 ; i-- )

        scanf("%d",&a[i]);

    printf("enter size of polynomial : ");

    scanf("%d",&n2);

    int b[n2];

    printf("enter the 2nd polynomial : ");

    for(int i=n2-1 ; i>=0 ; i-- )

        scanf("%d",&b[i]);

    int c[(n1+n2-1)];

    for(int i=0 ; i<n1+n1-1 ; i++)

    c[i]=0;

    for(int i=0 ; i<n1 ; i++){

        for(int j=0 ; j<n2 ; j++)

        c[i+j] += a[i]\*b[j];

    }

    printf("\n\nthe 1st polynomial is : ");

    for(int i=n1-1 ; i>=0 ; i-- )

        printf("%dx^%d ",a[i],i);

    printf("\n\nthe 2nd polynomial is : ");

    for(int i=n2-1 ; i>=0 ; i-- )

        printf("%dx^%d ",b[i],i);

    printf("\n\nthe polynomial is : ");

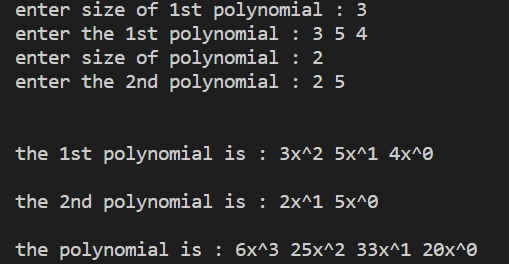
    for(int i=n1+n2-2 ; i>=0 ; i-- )

        printf("%dx^%d ",c[i],i);

    return 0;

}

**OUTPUT**



**Q8. Write a program to perform multiplication of two sparse matrices in triplet form.**

#include <stdio.h>

int main (){

    int m1 , n1 , m2 , n2 , nz1 , nz2 , k=1;

    printf("\n\nenter order and no. of non-zero element of matrix 1 : ");

    scanf("%d%d%d",&m1,&n1,&nz1);

    printf("\n\nenter order of matrix 2 : ");

    scanf("%d%d%d",&m2,&n2,&nz2);

    int a[nz1+1][3] , b[nz2+1][3];

    a[0][0] = m1 , a[0][1] = n1 , a[0][2] = nz1 , b[0][0] = m2 , b[0][1] = n2 , b[0][2] = nz2;

    printf("\n\n-------------------enter the triplet form---------------------");

    printf("\n\nmatrix 1 : \n");

    for(int i=1 ; i<=nz1 ; i++){

        printf("enter the detail for %d non-zero elemet : ",i);

        for(int j=0 ; j<3 ; j++)

        scanf("%d",&a[i][j]);

    }

    printf("\n\nmatrix 2 : \n");

    for(int i=1 ; i<=nz2 ; i++){

        printf("enter the detail for %d non-zero elemet : ",i);

        for(int j=0 ; j<3 ; j++)

        scanf("%d",&b[i][j]);

    }

    printf("\n\n-------------------the triplet form---------------------");

    printf("\n\nmatrix 1 : \n");

    for(int i=0 ; i<=nz1 ; i++){

        printf("\n");

        for(int j=0 ; j<3 ; j++)

        printf("%3d ",a[i][j]);

    }

    printf("\n\nmatrix 2 : \n");

    for(int i=0 ; i<=nz2 ; i++){

        printf("\n");

        for(int j=0 ; j<3 ; j++)

        printf("%3d ",b[i][j]);

    }

    int c[m1+m2+1][3];

    c[0][0] = a[0][0];

    c[0][1] = b[0][1];

    for(int i=1 ; i<=nz1 ; i++){

        for(int j=1; j<=nz2 ; j++){

            if(a[i][1] == b[j][0] ){            //for working with transpose b[0][1]

                c[k][0] = a[i][0];

                c[k][1] = b[j][1];              //for working with transpose b[0][0]

                c[k++][2] = a[i][2]\*b[j][2];

            }

        }

    }

    //sorting product array

    for(int j=1 ; j<k ; j++){

        for(int i=1 ; i<k-1 ; i++){

            if((c[i][1] > c[i+1][1]) && (c[i][0] == c[i+1][0])){

                for(int j=0 ; j<3 ; j++){

                    int temp = c[i][j];

                    c[i][j] = c[i+1][j];

                    c[i+1][j] = temp;

                }

            }

        }

    }

    // adding multiple entry of same elemement

    int i=1;

    while(i<k){

        int j = i+1;

        while(j<k){

            if((c[i][0] == c[j][0]) && (c[i][1] == c[j][1])){

                c[i][2] += c[j][2];

                for(int l=j ; l<k ; l++){

                    c[l][0] = c[l+1][0];

                    c[l][1] = c[l+1][1];

                    c[l][2] = c[l+1][2];

                }

                k--;

            }

            j++;

        }

        i++;

    }

    c[0][2] = k-1;

    printf("\n\ntriplet form product matrix :\n");

    for(int i=0 ; i<k ; i++){

        printf("\n");

        for(int j=0 ; j<3 ; j++)

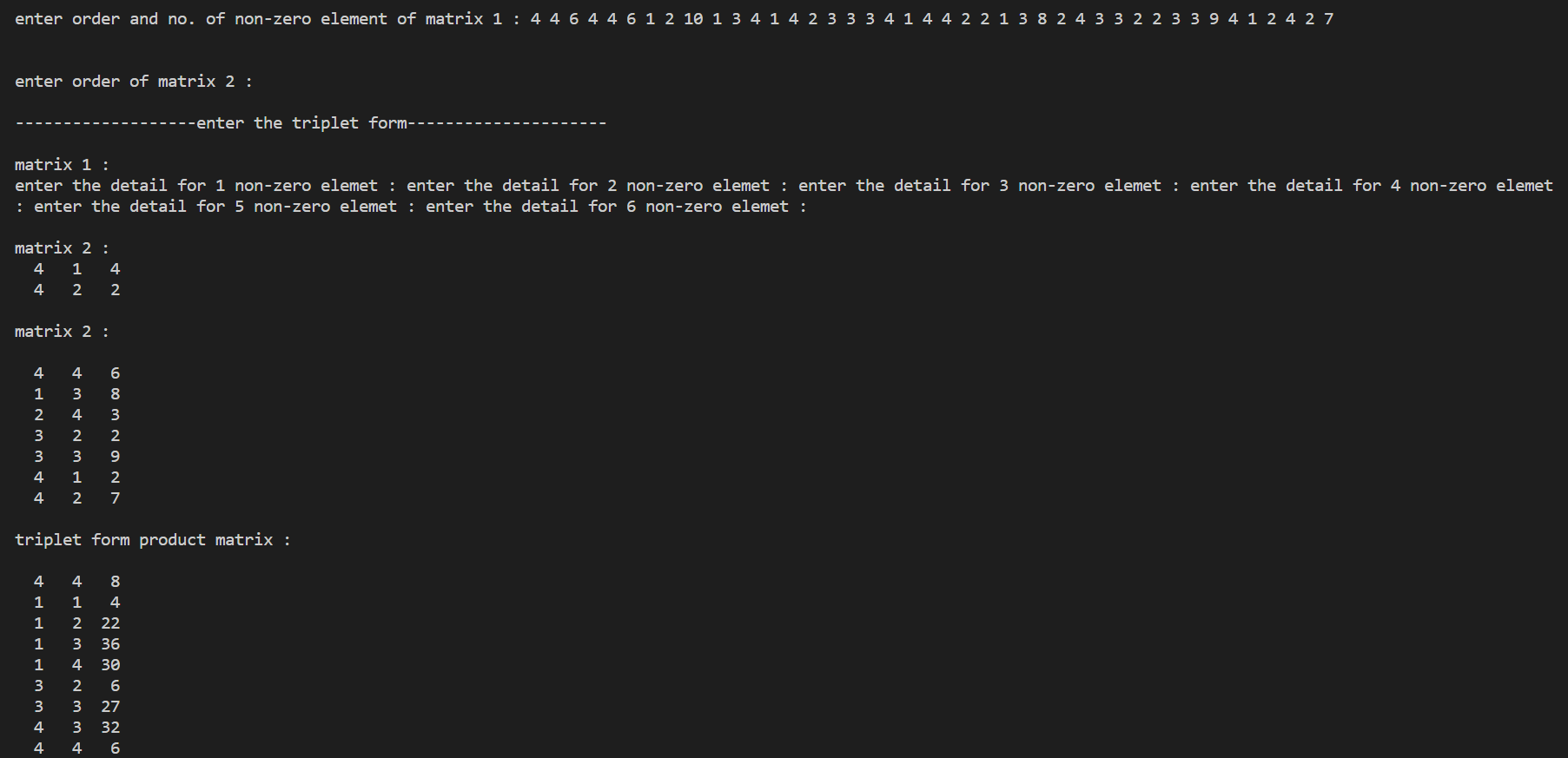
        printf("%3d ",c[i][j]);

    }

    return 0;

}

**OUTPUT**



**Q9. Write a program to create a polynomial with two variables and display it.**

#include <stdio.h>

#include <stdlib.h>

struct node{

    int coff;

    int x;

    int y;

    struct node \*next;

};

struct node\* cr(){

    struct node \*head = NULL , \*ptr , \*newnode;

    int n;

    printf("enter no. of terms in your polynominal : ");

    scanf("%d",&n);

    for(int i=0 ; i<n ; i++){

        printf("\n");

        newnode = (struct node\*)malloc(sizeof(struct node));

        printf("enter the coff of x and y : ");

        scanf("%d",&newnode->coff);

        printf("enter the power of x and y respectively : ");

        scanf("%d%d",&newnode->x , &newnode->y);

        newnode->next = NULL;

        if(head == NULL)

            head = ptr = newnode;

        else

            ptr = ptr->next = newnode;

    }

    return head;

}

void dis(struct node \*ptr){

    if(ptr == NULL){

        printf("\n\nthe list is empty");

        return;

    }

    while (ptr != NULL){

        printf("+%dx^%dy^%d ",ptr->coff , ptr->x , ptr->y);

        ptr = ptr->next;

    }

}

int main(){

    printf("\n\nenter your polynomial\n");

    struct node \*head = NULL;

    head = cr();

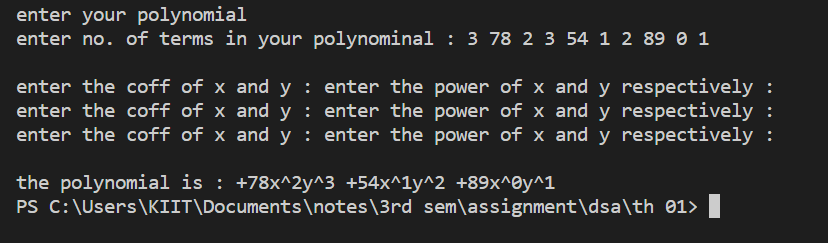
    printf("\n\nthe polynomial is : ");

    dis(head);

    return 0;

}

**OUTPUT**



**Q10. Find the middle element of the linked list without calculating the length.**

#include <stdio.h>

#include <stdlib.h>

struct node{

    int data;

    struct node \*next;

};

struct node\* cr(){

    struct node \*head = NULL , \*ptr , \*newnode;

    int n;

    printf("enter no. of node : ");

    scanf("%d",&n);

    for(int i=0 ; i<n ; i++){

        newnode = (struct node\*)malloc(sizeof(struct node));

        printf("enter data : ");

        scanf("%d",&newnode->data);

        newnode->next = NULL;

        if(head == NULL)

        head = ptr = newnode;

        else

        ptr = ptr->next = newnode;

    }

    return head;

};

void dis(struct node \*ptr){

    if(ptr == NULL){

        printf("\n\nthe list is empty");

        return;

    }

    printf("\nthe list is : ");

    while(ptr != NULL){

        printf("%d ",ptr->data);

        ptr = ptr->next;

    }

}

void mid(struct node \*ptr){

    if(ptr == NULL){

        printf("\n\nthe list is empty");

        return;

    }

    struct node \*temp = ptr;

    while((ptr != NULL) && (ptr->next != NULL)){

        temp = temp->next;

        ptr = ptr->next->next;

    }

    printf("\n\nthe middle node is : %d",temp->data);

}

int main(){

    struct node \*head = NULL;

    printf("\ncreate your list :-\n");

    head = cr();

    dis(head);

    mid(head);

    return 0;

}

**OUTPUT**

