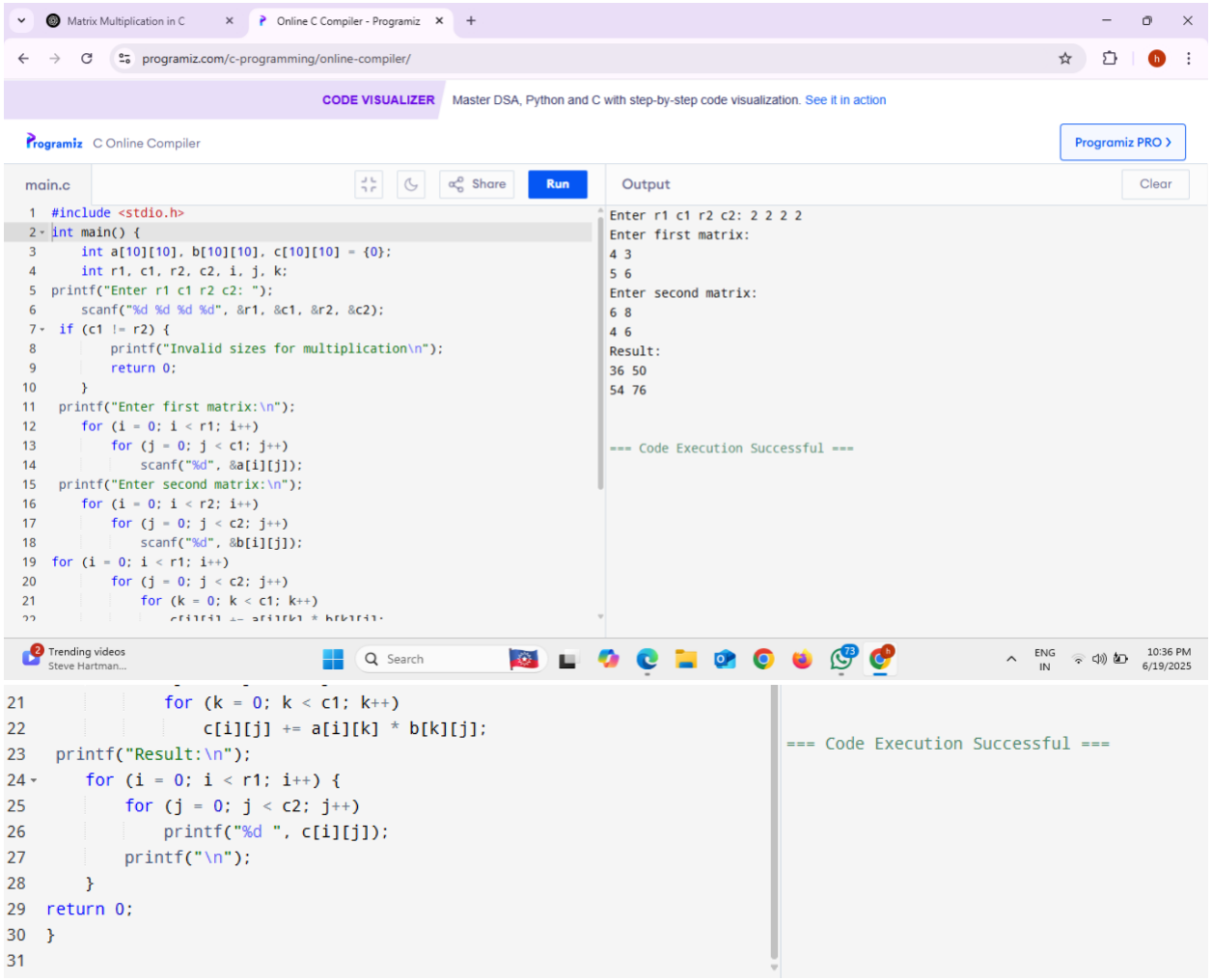
1 Write a C program to perform Matrix Multiplication.




```
            for (k = 0; k < c1; k++)
                c[i][j] += a[i][k] * b[k][j];
    printf("Result:\n");
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c2; j++)
            printf("%d ", c[i][j]);
        printf("\n");
    }
return 0;
}
```

```
=== Code Execution Successful ===
```

Result : code execution successful.

2. Write a C program to find Fibonacci series without using Recursion.

```c
#include <stdio.h>
int main() {
    int n, i;
    int a = 0, b = 1, c;
printf("Enter the number of terms: ");
    scanf("%d", &n);
 printf("Fibonacci Series: ");
    for (i = 0; i < n; i++) {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }
return 0;
}
```

```
Enter the number of terms: 7
Fibonacci Series: 0 1 1 2 3 5 8

=== Code Execution Successful ===
```

Result : code execution successful.

3. Write a C program to find Factorial of a given number using Recursion.

```c
#include <stdio.h>
int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
 if (num < 0)
        printf("Factorial not defined for negative numbers.\n");
    else
        printf("Factorial of %d is %d\n", num, factorial(num));
 return 0;
}
```

```
Enter a number: 5
Factorial of 5 is 120


=== Code Execution Successful ===
```

Result : code execution successful.

4. Write a C program to find Fibonacci series using Recursion.

```c
#include <stdio.h>
int fibonacci(int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}
int main() {
    int n, i;
  printf("Enter the number of terms: ");
    scanf("%d", &n);
  printf("Fibonacci Series: ");
    for (i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }
return 0;
}
```

```
Enter the number of terms: 7
Fibonacci Series: 0 1 1 2 3 5 8

=== Code Execution Successful ===
```

Result : code execution successful.

5. Write a C program to implement Array operations such as Insert, Delete and display.

```c
#include <stdio.h>
int main() {
    int a[100], n = 0, ch, i, pos, val;
do {
        printf("\n1.Insert 2.Delete 3.Display 4.Exit\nChoice: ");
        scanf("%d", &ch);
if (ch == 1) {
            printf("Pos & Val: ");
            scanf("%d%d", &pos, &val);
            for (i = n; i > pos; i--) a[i] = a[i - 1];
            a[pos] = val;
            n++;
        } else if (ch == 2) {
            printf("Pos: ");
            scanf("%d", &pos);
            for (i = pos; i < n - 1; i++) a[i] = a[i + 1];
            n--;
        } else if (ch == 3) {
            for (i = 0; i < n; i++) printf("%d ", a[i]);
        }} while (ch != 4);
 return 0;
```

```
1.Insert 2.Delete 3.Display 4.Exit
Choice:
1
Pos & Val: 0 10

1.Insert 2.Delete 3.Display 4.Exit
Choice: 2
Pos: 10

1.Insert 2.Delete 3.Display 4.Exit
Choice: 3

1.Insert 2.Delete 3.Display 4.Exit
Choice: 3

1.Insert 2.Delete 3.Display 4.Exit
Choice: 2
2
Pos:
1.Insert 2.Delete 3.Display 4.Exit
```

Result : code execution successful.

6. Write a C program to search a number using Linear Search method.

```c
#include <stdio.h>
int main() {
    int a[100], n, i, key, found = 0;
printf("Enter number of elements: ");
    scanf("%d", &n);
printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
 printf("Enter number to search: ");
    scanf("%d", &key);
 for (i = 0; i < n; i++) {
        if (a[i] == key) {
            printf("Number found at position %d\n", i);
            found = 1;
            break;
        } }
 if (!found)
        printf("Number not found\n");
return 0;
}
```

```
Enter number of elements: 5
Enter 5 elements:
2 3 6 3 5
Enter number to search: 5
Number found at position 4


=== Code Execution Successful ===
```

Result : code execution successful.

7. Write a C program to search a number using Binary Search method.

```c
int main() {
    int a[100], n, i, key, low, high, mid;
 printf("Enter number of elements: ");
    scanf("%d", &n);
printf("Enter %d sorted elements:\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
 printf("Enter number to search: ");
    scanf("%d", &key);
low = 0;
    high = n - 1;
 while (low <= high) {
        mid = (low + high) / 2;
        if (a[mid] == key) {
            printf("Number found at position %d\n", mid);
            return 0;
        } else if (a[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }} printf("Number not found\n");
    return 0;
```

```
Enter number of elements: 4
Enter 4 sorted elements:
20 30 40 50
Enter number to search: 30
Number found at position 1


=== Code Execution Successful ===
```

Result : code execution successful.

8 . Write a C program to implement Linked list operations.

```c
#include <stdio.h>
#include <stdlib.h>
struct N {
    int d;
    struct N *n;
} *head = NULL;
void insert(int v) {
    struct N *t = malloc(sizeof(struct N)), *p = head;
    t->d = v; t->n = NULL;
    if (!head) head = t;
    else {
        while (p->n) p = p->n;
        p->n = t;
    }}void del(int v) {
    struct N *t = head, *p = NULL;
    while (t && t->d != v) p = t, t = t->n;
    if (!t) return;
    if (!p) head = head->n;
    else p->n = t->n;
    free(t);}
void display() {
```

```
1.Insert 2.Delete 3.Display 4.Exit: 1
3
1.Insert 2.Delete 3.Display 4.Exit: 2
20
1.Insert 2.Delete 3.Display 4.Exit: 3
3
1.Insert 2.Delete 3.Display 4.Exit:
4


=== Code Execution Successful ===
```

```c
void display() {
    struct N *t = head;
    while (t) printf("%d ", t->d), t = t->n;
    printf("\n");}
int main() {
    int c, v;
    do {
        printf("1.Insert 2.Delete 3.Display 4.Exit: ");
        scanf("%d", &c);
        if (c == 1) scanf("%d", &v), insert(v);
        else if (c == 2) scanf("%d", &v), del(v);
        else if (c == 3) display();
    } while (c != 4);
    return 0;
}
```

```
1.Insert 2.Delete 3.Display 4.Exit:
4


=== Code Execution Successful ===
```

Result : code execution successful.

9. Write a C program to implement Stack operation such as PUSH,POP and PEEK.

```c
#include <stdio.h>
#define SIZE 100
int stack[SIZE], top = -1;
void push(int v) {
    if (top < SIZE - 1) stack[++top] = v;
    else printf("Overflow\n");
}void pop() {
    if (top >= 0) printf("Popped: %d\n", stack[top--]);
    else printf("Underflow\n");
}void peek() {
    if (top >= 0) printf("Top: %d\n", stack[top]);
    else printf("Empty\n");
}int main() {
    int ch, v;
    do {
        printf("1.Push 2.Pop 3.Peek 4.Exit: ");
        scanf("%d", &ch);
        if (ch == 1) scanf("%d", &v), push(v);
        else if (ch == 2) pop();
        else if (ch == 3) peek();
    } while (ch != 4);
```

```
1.Push 2.Pop 3.Peek 4.Exit: 1
10
1.Push 2.Pop 3.Peek 4.Exit: 2
Popped: 10
1.Push 2.Pop 3.Peek 4.Exit: 3
Empty
1.Push 2.Pop 3.Peek 4.Exit: 4


=== Code Execution Successful ===
```

Result : code execution successful.

10. Write a C program to implement the application of Stack(Notations).

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
char s[100]; int top = -1;
void push(char c) { s[++top] = c; }
char pop() { return s[top--]; }
int prec(char c) {
    if (c == '^') return 3;
    if (c == '*' || c == '/') return 2;
    if (c == '+' || c == '-') return 1;
    return 0;
}int main() {
    char in[100], out[100]; int i, j = 0;
    printf("Infix: "); scanf("%s", in);
for (i = 0; in[i]; i++) {
        if (isalnum(in[i])) out[j++] = in[i];
        else if (in[i] == '(') push(in[i]);
        else if (in[i] == ')') {
            while (s[top] != '(') out[j++] = pop();
            pop();
        } else {
        } else {
            while (top != -1 && prec(s[top]) >= prec(in[i]))
                out[j++] = pop();
            push(in[i]);
        } }while (top != -1) out[j++] = pop();
    out[j] = '\0';
printf("Postfix: %s\n", out);
    return 0;
}
```

```
Infix: a+b*c
Postfix: abc*+


=== Code Execution Successful ===
```

Result : code execution successful.

11. Write a C program to implement Queueo perations such as ENQUEUE,DEQUEUE and Display.

```c
#include <stdio.h>
#define S 100
int q[S], f = -1, r = -1;
void enq(int v) {
    if (r == S - 1) printf("Overflow\n");
    else {
        if (f == -1) f = 0;
        q[++r] = v;
    }}void deq() {
    if (f == -1 || f > r) printf("Underflow\n");
    else printf("Dequeued: %d\n", q[f++]);
}void dis() {
    if (f == -1 || f > r) printf("Empty\n");
    else for (int i = f; i <= r; i++) printf("%d ", q[i]);
}int main() {
    int c, v;
    do {
        printf("\n1.Enq 2.Deq 3.Show 4.Exit: ");
        scanf("%d", &c);
        if (c == 1) scanf("%d", &v), enq(v);
        else if (c == 2) deq();
          else if (c == 3) dis();
     } while (c != 4);
}
```

```
1.Enq 2.Deq 3.Show 4.Exit: 2
Underflow

1.Enq 2.Deq 3.Show 4.Exit: 1
10

1.Enq 2.Deq 3.Show 4.Exit: 2
Dequeued: 10

1.Enq 2.Deq 3.Show 4.Exit:
```

Result : code execution successful.

12. Write a C program to implement the Tree Traversals (Inorder, Preorder and postorder).

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *left, *right;
};
struct Node* create(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}void preorder(struct Node* root) {
    if (root) {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }}void inorder(struct Node* root) {
    if (root) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }}void postorder(struct Node* root) {
    if (root) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }}int main() {
    struct Node* root = create(1);
    root->left = create(2);
    root->right = create(3);
    root->left->left = create(4);
    root->left->right = create(5);
  printf("Preorder: ");
    preorder(root);
    printf("\nInorder: ");
    inorder(root);
    printf("\nPostorder: ");
    postorder(root);
  return 0;
}
```

```
Preorder: 1 2 4 5 3
Inorder: 4 2 5 1 3
Postorder: 4 5 2 3 1

=== Code Execution Successful ===
```

```
=== Code Execution Successful ===
```

Result : code execution successful.

13. Write a C program to implement hashing using Linear Probing method.

```c
#include <stdio.h>
#define SIZE 10
int hashTable[SIZE];
void init() {
    for (int i = 0; i < SIZE; i++)
        hashTable[i] = -1;
}int hash(int key) {
    return key % SIZE;
}void insert(int key) {
    int idx = hash(key);
    int i = 0;
while (hashTable[(idx + i) % SIZE] != -1 && i < SIZE)
        i++;
if (i < SIZE)
        hashTable[(idx + i) % SIZE] = key;
    else
        printf("Hash table is full!\n");
}
void display() {
    printf("Hash Table:\n");
    for (int i = 0; i < SIZE; i++)
        printf("[%d] => %d\n", i, hashTable[i]);
}
int main() {
    int choice, val;
init();
do {
        printf("\n1.Insert 2.Display 3.Exit\nChoice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &val);
                insert(val);
                break;
            case 2:
                display();
                break;
        }} while (choice != 3);
 return 0;
}
```

```
1.Insert 2.Display 3.Exit
Choice: 1
Enter value to insert: 3

1.Insert 2.Display 3.Exit
Choice: 2
Hash Table:
[0] => -1
[1] => -1
[2] => -1
[3] => 3
[4] => -1
[5] => -1
[6] => -1
[7] => -1
[8] => -1
[9] => -1

1.Insert 2.Display 3.Exit
Choice:
```

Result : code execution successful.

14. Write a C program to arrange a series of numbers using Insertion Sort.

```c
#include <stdio.h>
void insertionSort(int a[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = a[i];
        j = i - 1;
        while (j >= 0 && a[j] > key) {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;
    }}
int main() {
    int a[100], n, i;
 printf("Enter number of elements: ");
    scanf("%d", &n);
  printf("Enter %d numbers:\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
 insertionSort(a, n);
  printf("Sorted array:\n");
```

```
Enter number of elements: 4
Enter 4 numbers:
4 3 7 5
Sorted array:
3 4 5 7

=== Code Execution Successful ===
```

Result : code execution successful.

15. Write a C program to arrange a series of numbers using Merge Sort.

```c
#include <stdio.h>
void merge(int a[], int l, int m, int r) {
    int i = l, j = m + 1, k = 0;
    int temp[100];
while (i <= m && j <= r)
        temp[k++] = (a[i] < a[j]) ? a[i++] : a[j++];
 while (i <= m) temp[k++] = a[i++];
    while (j <= r) temp[k++] = a[j++];
for (i = l, k = 0; i <= r; i++, k++)
        a[i] = temp[k];
}void mergeSort(int a[], int l, int r) {
    if (l < r) {
        int m = (l + r) / 2;
        mergeSort(a, l, m);
        mergeSort(a, m + 1, r);
        merge(a, l, m, r);
    }}int main() {
    int a[100], n, i;
 printf("Enter number of elements: ");
    scanf("%d", &n);
 printf("Enter %d numbers:\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
mergeSort(a, 0, n - 1);
 printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    return 0;
}
```

```
Enter number of elements: 5
Enter 5 numbers:
3 5 8 9 7
Sorted array:
3 5 7 8 9

=== Code Execution Successful ===
```

Result : code execution successful.