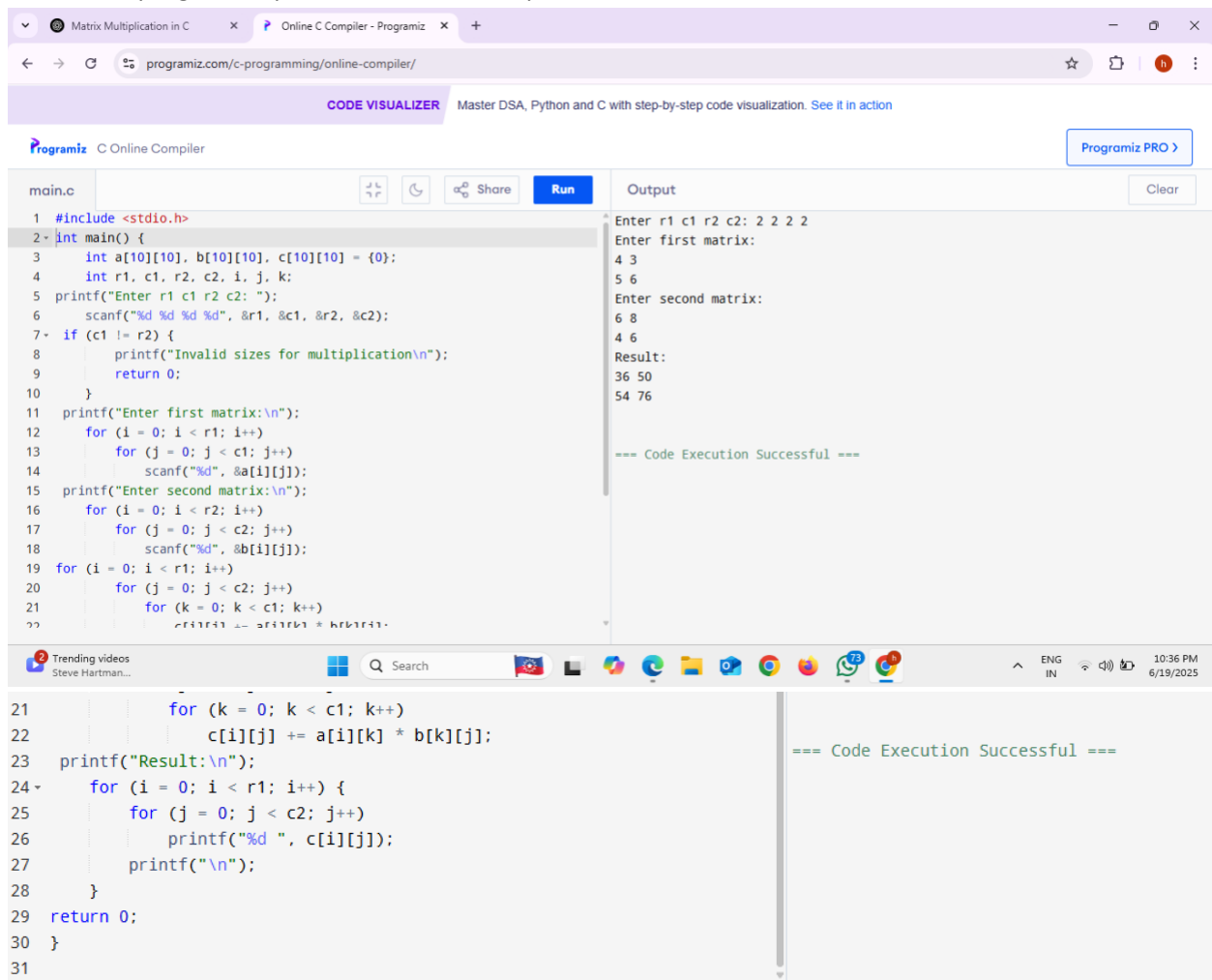


1 Write a C program to perform Matrix Multiplication.



The screenshot displays the Programiz Online C Compiler interface. The code editor on the left contains a C program for matrix multiplication. The program prompts the user to enter dimensions r1, c1, r2, and c2. It then prompts for the first matrix (r1 rows, c1 columns) and the second matrix (r2 rows, c2 columns). It checks if the multiplication is valid (c1 == r2). If valid, it calculates the result matrix (r1 rows, c2 columns) and prints it. The output window on the right shows the program's execution, including the input values, the matrices entered, and the resulting matrix.

```
1 #include <stdio.h>
2 int main() {
3     int a[10][10], b[10][10], c[10][10] = {0};
4     int r1, c1, r2, c2, i, j, k;
5     printf("Enter r1 c1 r2 c2: ");
6     scanf("%d %d %d %d", &r1, &c1, &r2, &c2);
7     if (c1 != r2) {
8         printf("Invalid sizes for multiplication\n");
9         return 0;
10    }
11    printf("Enter first matrix:\n");
12    for (i = 0; i < r1; i++)
13        for (j = 0; j < c1; j++)
14            scanf("%d", &a[i][j]);
15    printf("Enter second matrix:\n");
16    for (i = 0; i < r2; i++)
17        for (j = 0; j < c2; j++)
18            scanf("%d", &b[i][j]);
19    for (i = 0; i < r1; i++)
20        for (j = 0; j < c2; j++)
21            for (k = 0; k < c1; k++)
22                c[i][j] += a[i][k] * b[k][j];
23    printf("Result:\n");
24    for (i = 0; i < r1; i++) {
25        for (j = 0; j < c2; j++)
26            printf("%d ", c[i][j]);
27        printf("\n");
28    }
29    return 0;
30 }
31
```

Output:

```
Enter r1 c1 r2 c2: 2 2 2 2
Enter first matrix:
4 3
5 6
Enter second matrix:
6 8
4 6
Result:
36 50
54 76

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

Result : code execution successful.

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

1 #include <stdio.h>	Enter the number of terms: 7
2 int main() {	Fibonacci Series: 0 1 1 2 3 5 8
3 int n, i;	
4 int a = 0, b = 1, c;	=== Code Execution Successful ===
5 printf("Enter the number of terms: ");	
6 scanf("%d", &n);	
7 printf("Fibonacci Series: ");	
8 for (i = 0; i < n; i++) {	
9 printf("%d ", a);	
10 c = a + b;	
11 a = b;	
12 b = c;	
13 }	
14 return 0;	
15 }	

Result : code execution successful.

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

1 #include <stdio.h>	Enter a number: 5
2 int factorial(int n) {	Factorial of 5 is 120
3 if (n == 0 n == 1)	
4 return 1;	
5 else	=== Code Execution Successful ===
6 return n * factorial(n - 1);	
7 }	
8 int main() {	
9 int num;	
10 printf("Enter a number: ");	
11 scanf("%d", &num);	
12 if (num < 0)	
13 printf("Factorial not defined for negative numbers.\n");	
14 else	
15 printf("Factorial of %d is %d\n", num, factorial(num));	
16 return 0;	
17 }	

Result : code execution successful.

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

<pre> 1 #include <stdio.h> 2 int fibonacci(int n) { 3 if (n == 0) 4 return 0; 5 else if (n == 1) 6 return 1; 7 else 8 return fibonacci(n - 1) + fibonacci(n - 2); 9 } 10 int main() { 11 int n, i; 12 printf("Enter the number of terms: "); 13 scanf("%d", &n); 14 printf("Fibonacci Series: "); 15 for (i = 0; i < n; i++) { 16 printf("%d ", fibonacci(i)); 17 } 18 return 0; 19 } </pre>	<pre> Enter the number of terms: 7 Fibonacci Series: 0 1 1 2 3 5 8 === Code Execution Successful === </pre>
---	--

Result : code execution successful.

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

<pre> 1 #include <stdio.h> 2 int main() { 3 int a[100], n = 0, ch, i, pos, val; 4 do { 5 printf("\n1.Insert 2.Delete 3.Display 4.Exit\nChoice: "); 6 scanf("%d", &ch); 7 if (ch == 1) { 8 printf("Pos & Val: "); 9 scanf("%d%d", &pos, &val); 10 for (i = n; i > pos; i--) a[i] = a[i - 1]; 11 a[pos] = val; 12 n++; 13 } else if (ch == 2) { 14 printf("Pos: "); 15 scanf("%d", &pos); 16 for (i = pos; i < n - 1; i++) a[i] = a[i + 1]; 17 n--; 18 } else if (ch == 3) { 19 for (i = 0; i < n; i++) printf("%d ", a[i]); 20 } while (ch != 4); 21 return 0; </pre>	<pre> 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 </pre>
---	--

Result : code execution successful.

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

<pre> 1 #include <stdio.h> 2 int main() { 3 int a[100], n, i, key, found = 0; 4 printf("Enter number of elements: "); 5 scanf("%d", &n); 6 printf("Enter %d elements:\n", n); 7 for (i = 0; i < n; i++) 8 scanf("%d", &a[i]); 9 printf("Enter number to search: "); 10 scanf("%d", &key); 11 for (i = 0; i < n; i++) { 12 if (a[i] == key) { 13 printf("Number found at position %d\n", i); 14 found = 1; 15 break; 16 } } 17 if (!found) 18 printf("Number not found\n"); 19 return 0; 20 } </pre>	<pre> 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 === </pre>
---	--

Result : code execution successful.

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

<pre> 2 int main() { 3 int a[100], n, i, key, low, high, mid; 4 printf("Enter number of elements: "); 5 scanf("%d", &n); 6 printf("Enter %d sorted elements:\n", n); 7 for (i = 0; i < n; i++) 8 scanf("%d", &a[i]); 9 printf("Enter number to search: "); 10 scanf("%d", &key); 11 low = 0; 12 high = n - 1; 13 while (low <= high) { 14 mid = (low + high) / 2; 15 if (a[mid] == key) { 16 printf("Number found at position %d\n", mid); 17 return 0; 18 } else if (a[mid] < key) { 19 low = mid + 1; 20 } else { 21 high = mid - 1; 22 } } printf("Number not found\n"); 23 return 0; </pre>	<pre> 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 === </pre>
--	--

Result : code execution successful.

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

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 struct N {
4     int d;
5     struct N *n;
6 } *head = NULL;
7 void insert(int v) {
8     struct N *t = malloc(sizeof(struct N)); *p = head;
9     t->d = v; t->n = NULL;
10    if (!head) head = t;
11    else {
12        while (p->n) p = p->n;
13        p->n = t;
14    } void del(int v) {
15        struct N *t = head, *p = NULL;
16        while (t && t->d != v) p = t, t = t->n;
17        if (!t) return;
18        if (!p) head = head->n;
19        else p->n = t->n;
20        free(t);
21 void display() {
22     struct N *t = head;
23     while (t) printf("%d ", t->d), t = t->n;
24     printf("\n");
25 int main() {
26     int c, v;
27     do {
28         printf("1.Insert 2.Delete 3.Display 4.Exit: ");
29         scanf("%d", &c);
30         if (c == 1) scanf("%d", &v), insert(v);
31         else if (c == 2) scanf("%d", &v), del(v);
32         else if (c == 3) display();
33     } while (c != 4);
34     return 0;
35 }

```

```

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 ===

```

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.

```

1 #include <stdio.h>
2 #define SIZE 100
3 int stack[SIZE], top = -1;
4 void push(int v) {
5     if (top < SIZE - 1) stack[++top] = v;
6     else printf("Overflow\n");
7 } void pop() {
8     if (top >= 0) printf("Popped: %d\n", stack[top--]);
9     else printf("Underflow\n");
10 } void peek() {
11     if (top >= 0) printf("Top: %d\n", stack[top]);
12     else printf("Empty\n");
13 } int main() {
14     int ch, v;
15     do {
16         printf("1.Push 2.Pop 3.Peek 4.Exit: ");
17         scanf("%d", &ch);
18         if (ch == 1) scanf("%d", &v), push(v);
19         else if (ch == 2) pop();
20         else if (ch == 3) peek();
21     } 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).

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <ctype.h>
4 char s[100]; int top = -1;
5 void push(char c) { s[++top] = c; }
6 char pop() { return s[top--]; }
7 int prec(char c) {
8     if (c == '^') return 3;
9     if (c == '*' || c == '/') return 2;
10    if (c == '+' || c == '-') return 1;
11    return 0;
12 }
13 int main() {
14     char in[100], out[100]; int i, j = 0;
15     printf("Infix: "); scanf("%s", in);
16     for (i = 0; in[i]; i++) {
17         if (isdigit(in[i])) out[j++] = in[i];
18         else if (in[i] == '(') push(in[i]);
19         else if (in[i] == ')') {
20             while (s[top] != '(') out[j++] = pop();
21             pop();
22         } else {
23             while (top != -1 && prec(s[top]) >= prec(in[i]))
24                 out[j++] = pop();
25             push(in[i]);
26         }
27     }
28     while (top != -1) out[j++] = pop();
29     out[j] = '\0';
30     printf("Postfix: %s\n", out);
31     return 0;
32 }

```

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

=== Code Execution Successful ===

Result : code execution successful.

11. Write a C program to implement Queue operations such as ENQUEUE, DEQUEUE and Display.

```

1 #include <stdio.h>
2 #define S 100
3 int q[S], f = -1, r = -1;
4 void enq(int v) {
5     if (r == S - 1) printf("Overflow\n");
6     else {
7         if (f == -1) f = 0;
8         q[++r] = v;
9     }
10 }
11 void deq() {
12     if (f == -1 || f > r) printf("Underflow\n");
13     else printf("Dequeued: %d\n", q[f++]);
14 }
15 void dis() {
16     if (f == -1 || f > r) printf("Empty\n");
17     else for (int i = f; i <= r; i++) printf("%d ", q[i]);
18 }
19 int main() {
20     int c, v;
21     do {
22         printf("\n1.Enq 2.Deq 3.Show 4.Exit: ");
23         scanf("%d", &c);
24         if (c == 1) scanf("%d", &v), enq(v);
25         else if (c == 2) deq();
26         else if (c == 3) dis();
27     } while (c != 4);
28 }

```

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).

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Node {
4     int data;
5     struct Node *left, *right;
6 };
7 struct Node* create(int value) {
8     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
9     newNode->data = value;
10    newNode->left = newNode->right = NULL;
11    return newNode;
12 }
13 void preorder(struct Node* root) {
14     if (root) {
15         printf("%d ", root->data);
16         preorder(root->left);
17         preorder(root->right);
18     }
19 }
20 void inorder(struct Node* root) {
21     if (root) {
22         inorder(root->left);
23         printf("%d ", root->data);
24         inorder(root->right);
25     }
26 }
27 void postorder(struct Node* root) {
28     if (root) {
29         postorder(root->left);
30         postorder(root->right);
31         printf("%d ", root->data);
32     }
33 }
34 int main() {
35     struct Node* root = create(1);
36     root->left = create(2);
37     root->right = create(3);
38     root->left->left = create(4);
39     root->left->right = create(5);
40     printf("Preorder: ");
41     preorder(root);
42     printf("\nInorder: ");
43     inorder(root);
44     printf("\nPostorder: ");
45     postorder(root);
46     return 0;
47 }
```

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.

```
1 #include <stdio.h>
2 #define SIZE 10
3 int hashTable[SIZE];
4 void init() {
5     for (int i = 0; i < SIZE; i++)
6         hashTable[i] = -1;
7 }
8 int hash(int key) {
9     return key % SIZE;
10 }
11 void insert(int key) {
12     int idx = hash(key);
13     int i = 0;
14     while (hashTable[(idx + i) % SIZE] != -1 && i < SIZE)
15         i++;
16     if (i < SIZE)
17         hashTable[(idx + i) % SIZE] = key;
18     else
19         printf("Hash table is full!\n");
20 }
21 void display() {
22     printf("Hash Table:\n");
23     for (int i = 0; i < SIZE; i++)
24         printf("[%d] => %d\n", i, hashTable[i]);
25 }
26 int main() {
27     int choice, val;
28     init();
29     do {
30         printf("\n1.Insert 2.Display 3.Exit\nChoice: ");
31         scanf("%d", &choice);
32         switch (choice) {
33             case 1:
34                 printf("Enter value to insert: ");
35                 scanf("%d", &val);
36                 insert(val);
37                 break;
38             case 2:
39                 display();
40                 break;
41             default:
42                 break;
43         }
44     } while (choice != 3);
45     return 0;
46 }
```

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.

```
1 #include <stdio.h>
2 void insertionSort(int a[], int n) {
3     int i, key, j;
4     for (i = 1; i < n; i++) {
5         key = a[i];
6         j = i - 1;
7         while (j >= 0 && a[j] > key) {
8             a[j + 1] = a[j];
9             j--;
10        }
11        a[j + 1] = key;
12    }
13}
14 int main() {
15     int a[100], n, i;
16     printf("Enter number of elements: ");
17     scanf("%d", &n);
18     printf("Enter %d numbers:\n", n);
19     for (i = 0; i < n; i++)
20         scanf("%d", &a[i]);
21     insertionSort(a, n);
22     printf("Sorted array:\n");
23 }
```

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.

```
1 #include <stdio.h>
2 void merge(int a[], int l, int m, int r) {
3     int i = l, j = m + 1, k = 0;
4     int temp[100];
5     while (i <= m && j <= r)
6         temp[k++] = (a[i] < a[j]) ? a[i++] : a[j++];
7     while (i <= m) temp[k++] = a[i++];
8     while (j <= r) temp[k++] = a[j++];
9     for (i = l, k = 0; i <= r; i++, k++)
10        a[i] = temp[k];
11}
12 void mergeSort(int a[], int l, int r) {
13     if (l < r) {
14         int m = (l + r) / 2;
15         mergeSort(a, l, m);
16         mergeSort(a, m + 1, r);
17         merge(a, l, m, r);
18     }
19 }
20 int main() {
21     int a[100], n, i;
22     printf("Enter number of elements: ");
23     scanf("%d", &n);
24     printf("Enter %d numbers:\n", n);
25     for (i = 0; i < n; i++)
26         scanf("%d", &a[i]);
27     mergeSort(a, 0, n - 1);
28     printf("Sorted array:\n");
29     for (i = 0; i < n; i++)
30         printf("%d ", a[i]);
31     return 0;
32 }
```

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.