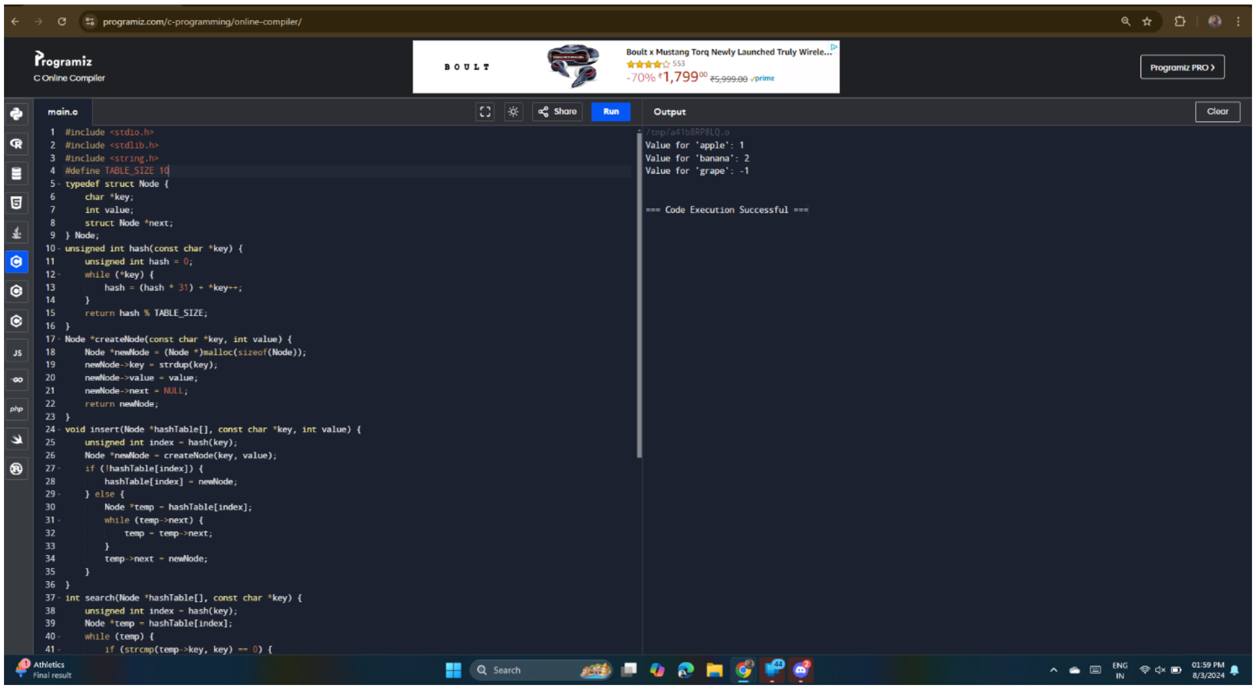
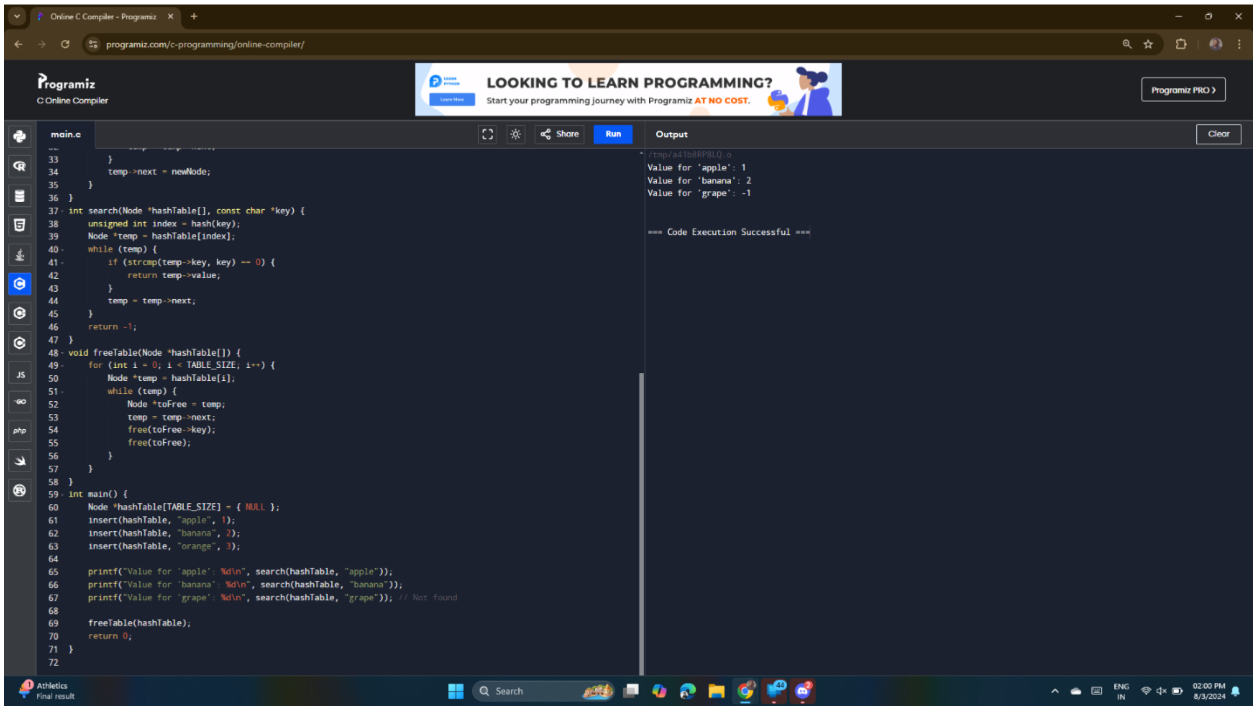
write a program of hashing of table



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TABLE_SIZE 10
typedef struct Node {
    char *key;
    int value;
    struct Node *next;
} Node;
unsigned int hash(const char *key) {
    unsigned int hash = 0;
    while (*key) {
        hash = (hash * 31) + *key++;
    }
    return hash % TABLE_SIZE;
}
Node *createNode(const char *key, int value) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->key = strdup(key);
    newNode->value = value;
    newNode->next = NULL;
    return newNode;
}
void insert(Node *hashTable[], const char *key, int value) {
    unsigned int index = hash(key);
    Node *newNode = createNode(key, value);
    if (!hashTable[index]) {
        hashTable[index] = newNode;
    } else {
        Node *temp = hashTable[index];
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
int search(Node *hashTable[], const char *key) {
    unsigned int index = hash(key);
    Node *temp = hashTable[index];
    while (temp) {
        if (strcmp(temp->key, key) == 0) {
```

```
Value for 'apple': 1
Value for 'banana': 2
Value for 'grape': -1


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



```c
        }
        temp->next = newNode;
    }
}
int search(Node *hashTable[], const char *key) {
    unsigned int index = hash(key);
    Node *temp = hashTable[index];
    while (temp) {
        if (strcmp(temp->key, key) == 0) {
            return temp->value;
        }
        temp = temp->next;
    }
    return -1;
}
void freeTable(Node *hashTable[]) {
    for (int i = 0; i < TABLE_SIZE; i++) {
        Node *temp = hashTable[i];
        while (temp) {
            Node *toFree = temp;
            temp = temp->next;
            free(toFree->key);
            free(toFree);
        }
    }
}
int main() {
    Node *hashTable[TABLE_SIZE] = { NULL };
    insert(hashTable, "apple", 1);
    insert(hashTable, "banana", 2);
    insert(hashTable, "orange", 3);

    printf("Value for 'apple': %d\n", search(hashTable, "apple"));
    printf("Value for 'banana': %d\n", search(hashTable, "banana"));
    printf("Value for 'grape': %d\n", search(hashTable, "grape")); // Not found

    freeTable(hashTable);
    return 0;
}
```

```
Value for 'apple': 1
Value for 'banana': 2
Value for 'grape': -1


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