

LAB ASSIGNMENT

DATA STRUCTURES & ALGORITHM

ARRAYS

1. Write a program to initialize an array of integers and print all the elements

CODE

```
#include <iostream>

using namespace std;

int main() {

    int arr[] = {1, 2, 3, 4, 5}; // Initializing array with 5 integers

    int length = sizeof(arr) / sizeof(arr[0]);

    for (int i = 0; i < length; i++) {

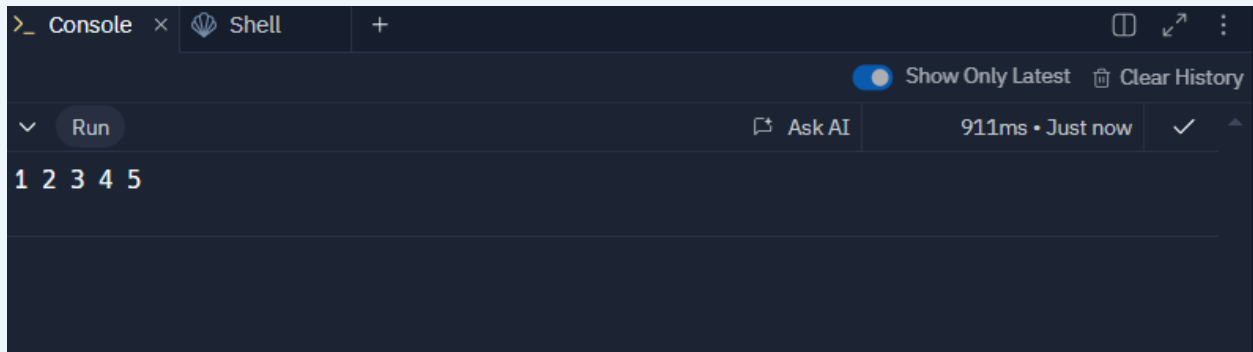
        cout << arr[i] << " ";

    }

    return 0;
```

```
}
```

OUTPUT



```
>_ Console x Shell +  
Show Only Latest Clear History  
Run Ask AI 911ms • Just now ✓  
1 2 3 4 5
```

2. Create a program that finds the maximum and minimum values in an array.

CODE

```
#include <iostream>
```

```
Using namespace std;
```

```
#include <algorithm> // For max and min
```

```
int main()
```

```
{
```

```
    int arr[] = {3, 6, 4, 1, 8, 9, 2, 6, 5, 3}; //Array initialization
```

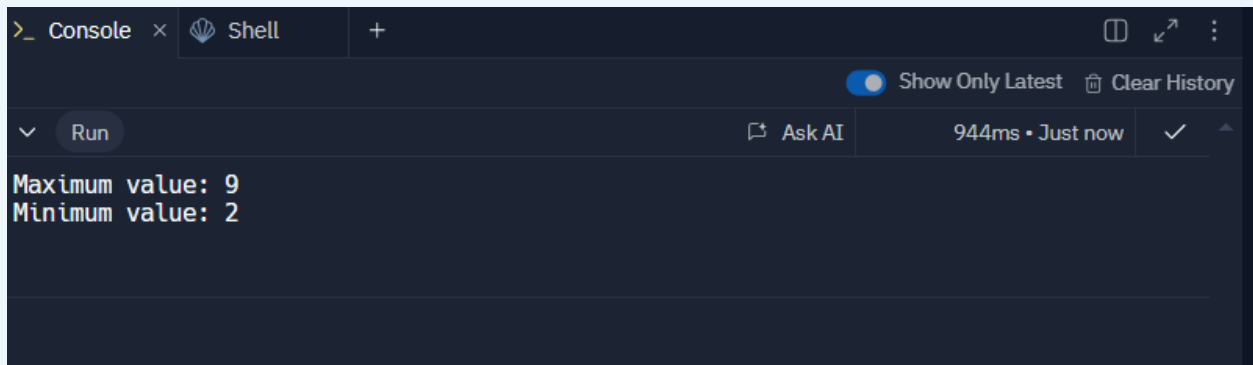
```
    int length = sizeof(arr) / sizeof(arr[0]);
```

```
    int maxVal = arr[0];
```

```
    int minVal = arr[0];
```

```
for (int i = 1; i < length; i++){  
    if (arr[i] > maxVal){  
        maxVal = arr[i];  
    }  
    if (arr[i] < minVal)  
{  
        minVal = arr[i];  
    }  
}  
  
cout << "Maximum Value: " << maxVal << endl;  
cout << "Minimum Value: " << minVal << endl;  
  
return 0;  
}
```

OUTPUT

A screenshot of a code editor's console window. The window has a dark theme. At the top, there are tabs for 'Console' and 'Shell'. Below the tabs, there's a toolbar with a 'Run' button, an 'Ask AI' button, and a 'Show Only Latest' toggle. The console output shows 'Maximum value: 9' and 'Minimum value: 2' on two separate lines. The output is in a light green monospace font. The console also shows a status bar at the bottom with '944ms • Just now' and a checkmark icon.

```
>_ Console x Shell +  
Show Only Latest Clear History  
Run Ask AI 944ms • Just now ✓  
Maximum value: 9  
Minimum value: 2
```

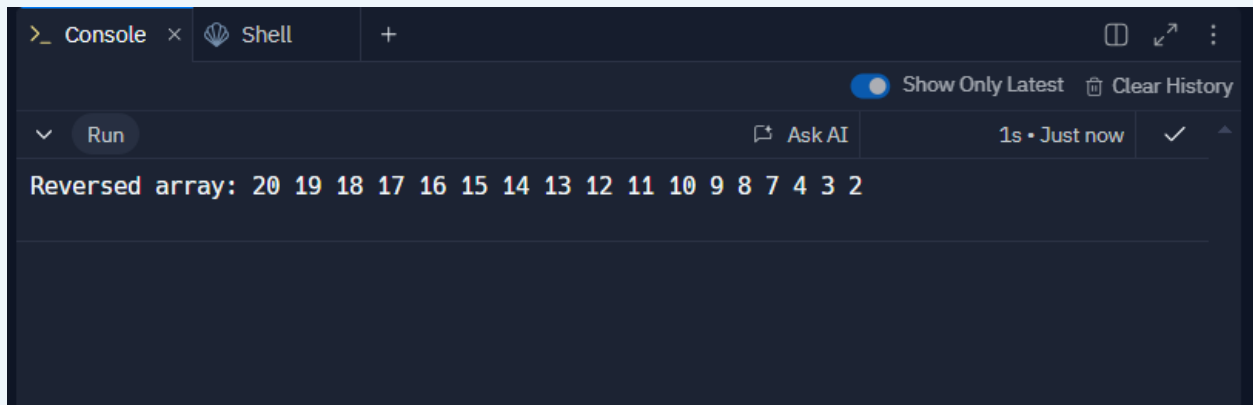
3. Write a program that reverses the elements of an array.

CODE

```
#include <iostream>  
  
using namespace std;  
  
#include <algorithm> // For reverse  
  
int main() {  
  
    int arr[] = {2,3,4,7,8,9,10,11,12,13,14,15,16,17,18,19,20};  
  
    int length = sizeof(arr) / sizeof(arr[0]);  
  
  
    reverse(arr, arr + length); // Reverse the array  
  
  
    cout << "Reversed array: ";  
  
    for (int i = 0; i < length; i++){  
  
        cout << arr[i] << " ";  
  
    }  
}
```

```
    return 0;  
}
```

OUTPUT

A screenshot of a code editor's console window. The window has tabs for 'Console' and 'Shell'. The 'Console' tab is active, showing a 'Run' button and an 'Ask AI' button. The output text reads 'Reversed array: 20 19 18 17 16 15 14 13 12 11 10 9 8 7 4 3 2'. Above the output, there are controls for 'Show Only Latest' (a toggle switch) and 'Clear History' (a trash icon). The output is displayed in a dark-themed interface with light-colored text.

```
>_ Console x Shell +  
Show Only Latest Clear History  
Run Ask AI 1s • Just now ✓  
Reversed array: 20 19 18 17 16 15 14 13 12 11 10 9 8 7 4 3 2
```

LISTS

1. Implement a program to add and display elements in a linked list.

CODE

```
#include <iostream>  
  
using namespace std;  
  
// Node structure  
  
struct Node  
{  
    int data;
```

```
Node* next;

};

// Function to add a new node at the end
void addNode(Node*& head, int value)
{
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;

    if (head == nullptr)
    {
        head = newNode;
    } else
    {
        Node* temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
```

```
}
```

```
// Function to display the linked list
```

```
void displayList(Node* head)
```

```
{
```

```
    Node* temp = head;
```

```
    while (temp != nullptr)
```

```
{
```

```
        cout << temp->data << " ";
```

```
        temp = temp->next;
```

```
}
```

```
    cout << endl;
```

```
}
```

```
int main() {
```

```
    Node* head = nullptr; // Initialize an empty list
```

```
    // Add elements to the list
```

```
    addNode(head, 4);
```

```
    addNode(head, 2);
```

```
    addNode(head, 5);
```

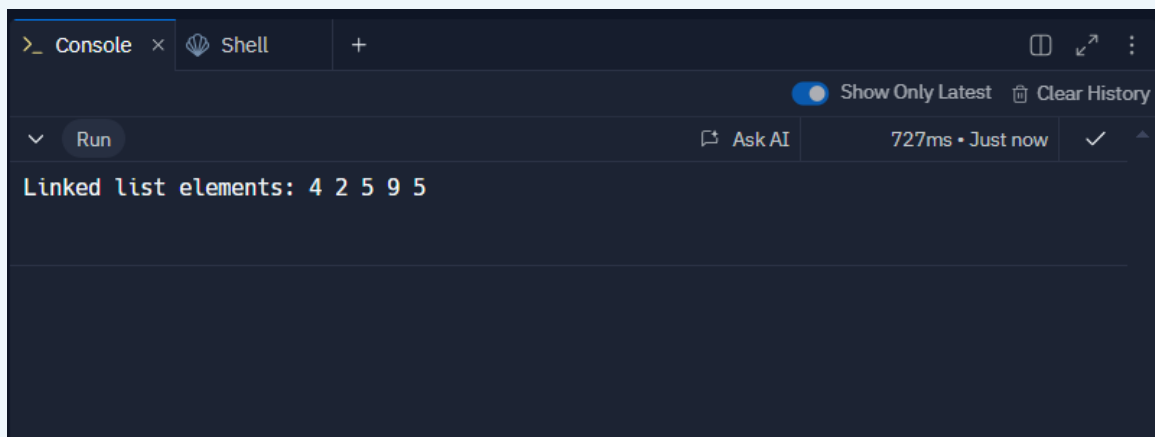
```
    addNode(head, 9);
```

```
addNode(head, 5);

// Display the list
cout << "Linked list elements: ";
displayList(head);

return 0;
}
```

OUTPUT

A screenshot of a code editor's console window. The window has a dark theme. At the top, there are tabs for 'Console' and 'Shell'. Below the tabs, there is a toolbar with a 'Run' button, an 'Ask AI' button, and a status bar showing '727ms • Just now'. The main area of the console displays the output 'Linked list elements: 4 2 5 9 5' in a monospaced font.

2. Implement a program to remove and display elements in a linked list.

CODE

```
#include <iostream>

using namespace std;

// Node structure
```



```
struct Node
```

```
{
```

```
    int data;
```

```
    Node* next;
```

```
};
```

```
// Function to add a new node at the end
```

```
void addNode(Node*& head, int value) {
```

```
    Node* newNode = new Node();
```

```
    newNode->data = value;
```

```
    newNode->next = nullptr;
```

```
    if (head == nullptr) {
```

```
        head = newNode;
```

```
    } else {
```

```
        Node* temp = head;
```

```
        while (temp->next != nullptr) {
```

```
            temp = temp->next;
```

```
        }
```

```
        temp->next = newNode;
```

```
    }
```

```
}
```

// Function to remove a node with a specific value

```
void removeNode(Node*& head, int value){
```

```
    if (head == nullptr) return;
```

```
    // If head needs to be removed
```

```
    if (head->data == value){
```

```
        Node* temp = head;
```

```
        head = head->next;
```

```
        delete temp;
```

```
        return;
```

```
    }
```

```
    Node* temp = head;
```

```
    while (temp->next != nullptr && temp->next->data != value){
```

```
        temp = temp->next;
```

```
    }
```

```
    if (temp->next == nullptr) return; // Value not found
```

```
    Node* nodeToDelete = temp->next;
```

```
    temp->next = temp->next->next;
```

```
        delete nodeToDelete;
    }

// Function to display the linked list
void displayList(Node* head){
    Node* temp = head;
    while (temp != nullptr){
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
```

```
int main(){
    Node* head = nullptr; /
    / Initialize an empty list

    // Add elements to the list
    addNode(head, 4);
    addNode(head, 5);
    addNode(head, 7);
    addNode(head, 8);
```

```
addNode(head, 9);

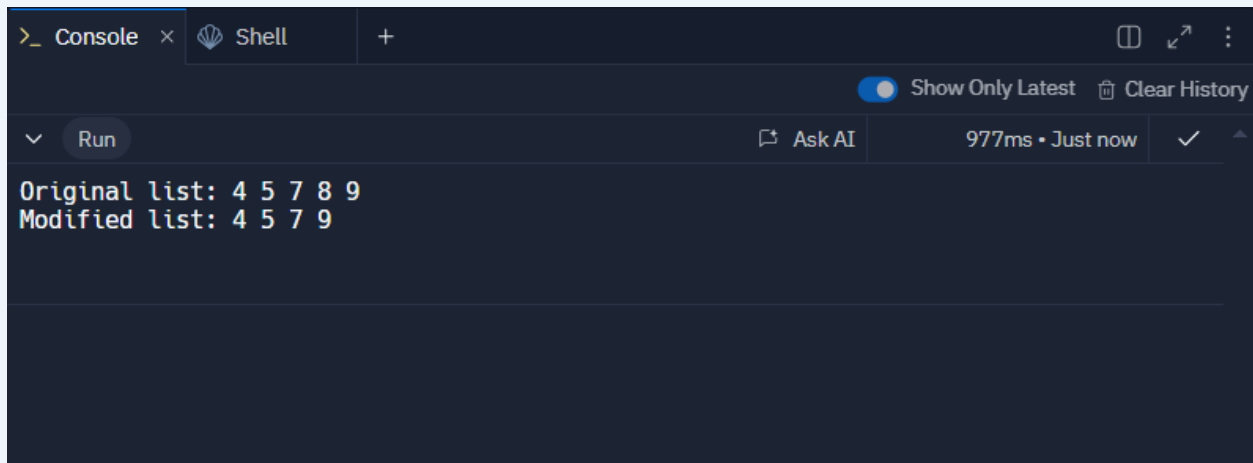
// Display the list
cout << "Original list: ";
displayList(head);

// Remove elements from the list
removeNode(head, 2);
removeNode(head, 8);

// Display the list after removal
cout << "Modified list: ";
displayList(head);

return 0;
}
```

OUTPUT



```
>_ Console x Shell +
Show Only Latest Clear History
Run Ask AI 977ms • Just now ✓
Original list: 4 5 7 8 9
Modified list: 4 5 7 9
```

3. Write a program that searches for an element in a linked list.

CODE

```
#include <iostream>

using namespace std;

// Node structure

struct Node

{

    int data;

    Node* next;

};

// Function to add a new node at the end

void addNode(Node*& head, int value) {

    Node* newNode = new Node();

    newNode->data = value;
```

```
newNode->next = nullptr;
```

```
if (head == nullptr) {
```

```
    head = newNode;
```

```
} else {
```

```
    Node* temp = head;
```

```
    while (temp->next != nullptr) {
```

```
        temp = temp->next;
```

```
    }
```

```
    temp->next = newNode;
```

```
}
```

```
}
```

```
// Function to search for an element
```

```
bool searchElement(Node* head, int value) {
```

```
    Node* temp = head;
```

```
    while (temp != nullptr) {
```

```
        if (temp->data == value) {
```

```
            return true;
```

```
        }
```

```
        temp = temp->next;
```

```
}
```

```
        return false;
    }

// Function to display the linked list
void displayList(Node* head)
{
    Node* temp = head;
    while (temp != nullptr)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    // Initialize an empty list

    // Add elements to the list
    addNode(head, 4);
    addNode(head, 5);
```

```
addNode(head, 6);

addNode(head, 8);

addNode(head, 0);


// Display the list

cout << "Linked list elements: ";

displayList(head);


// Search for an element

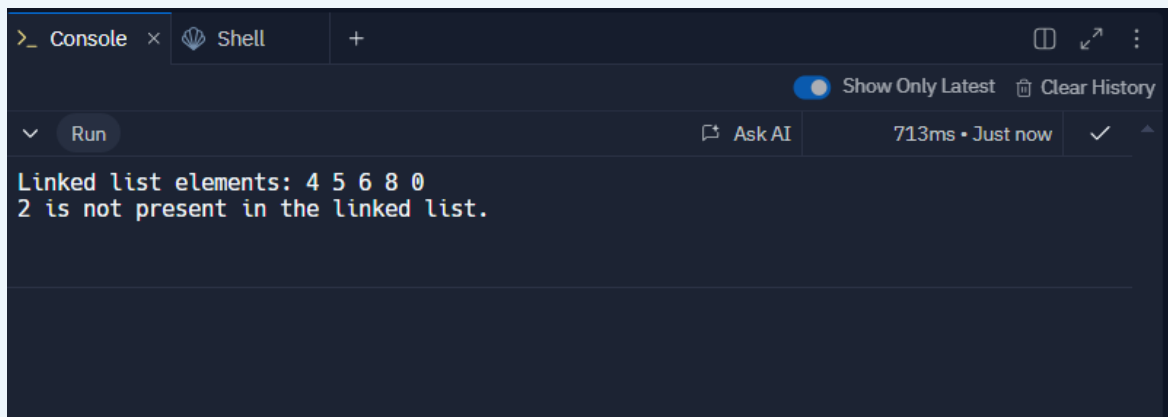
int searchValue = 2;

if (searchElement(head, searchValue))
{
    cout << searchValue << " is present in the linked list." << endl;
} else {
    cout << searchValue << " is not present in the linked list." << endl;
}


return 0;

}
```

OUTPUT

A screenshot of a code editor's console window. The window has a dark theme. At the top, there are tabs for 'Console' and 'Shell'. Below the tabs, there's a status bar with 'Show Only Latest' and 'Clear History' buttons. The main area of the console shows the output of a program: 'Linked list elements: 4 5 6 8 0' and '2 is not present in the linked list.' The text is in a monospaced font, with the first line in white and the second line in a light blue color. There are also some UI elements like a 'Run' button, 'Ask AI' button, and a timer '713ms • Just now'.

4. Create a program to insert an element at the beginning of linked list.

CODE

```
#include <iostream>
```

```
Using namespace std;
```

```
// Node structure
```

```
struct Node {
```

```
    int data;
```

```
    Node* next;
```

```
};
```

```
// Function to add a new node at the beginning
```

```
void addNodeAtBeginning(Node*& head, int value) {
```

```
    Node* newNode = new Node();
```

```
    newNode->data = value;
```

```
    newNode->next = head;
```

```
    head = newNode;
}
```

```
// Function to display the linked list
```

```
void displayList(Node* head){
    Node* temp = head;
    while (temp != nullptr){
        cout << temp->data << " ";
        temp = temp->next;
    }
    std::cout << std::endl;
}
```

```
int main(){
    Node* head = nullptr; // Initialize an empty list

    // Add elements to the list
    addNodeAtBeginning(head, 4);
    addNodeAtBeginning(head, 5);
    addNodeAtBeginning(head, 6);
    addNodeAtBeginning(head, 7);
    addNodeAtBeginning(head, 8);
}
```

```
// Display the list

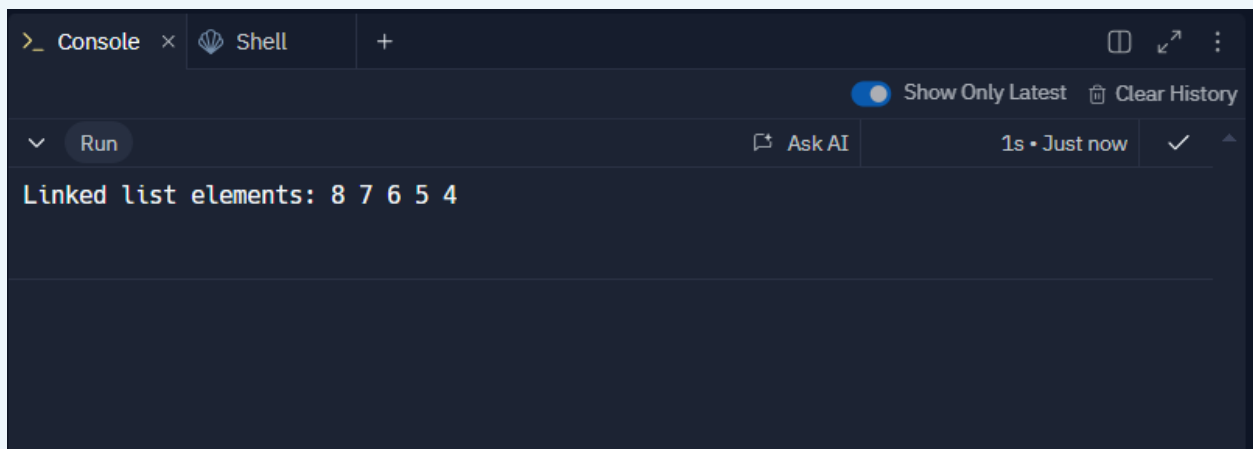
std::cout << "Linked list elements: ";

displayList(head);

return 0;

}
```

OUTPUT

A screenshot of a code editor's console window. The window has a dark background with a light blue header bar. The header bar contains tabs for 'Console' (selected) and 'Shell', along with a '+' icon. On the right side of the header bar, there are icons for a window, a refresh arrow, and a vertical ellipsis. Below the header bar, there is a toolbar with a 'Run' button, an 'Ask AI' button, and a status bar showing '1s • Just now' and a checkmark. The main area of the console displays the output 'Linked list elements: 8 7 6 5 4' in a monospaced font.

5. Create a program to insert an element at the middle of linked list.

CODE

```
#include <iostream>

using namespace std;

// Node structure
```

```
struct Node
```

```
{
```

```
    int data;
```

```
    Node* next;
```

```
};
```

```
// Function to add a new node at the middle
```

```
void addNodeAtMiddle(Node*& head, int value)
```

```
{
```

```
    Node* newNode = new Node();
```

```
    newNode->data = value;
```

```
    if (head == nullptr || head->next == nullptr)
```

```
{
```

```
        newNode->next = head;
```

```
        head = newNode;
```

```
        return;
```

```
}
```

```
Node* slow = head;
```

```
Node* fast = head;
```

```
while (fast->next != nullptr && fast->next->next != nullptr)
{
    slow = slow->next;
    fast = fast->next->next;
}
```

```
newNode->next = slow->next;
slow->next = newNode;
}
```

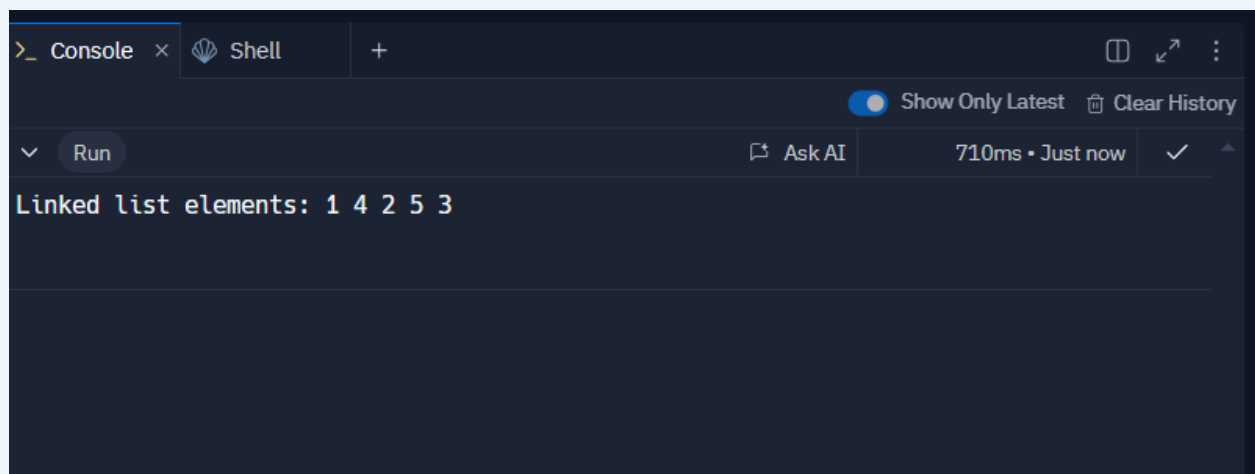
// Function to display the linked list

```
void displayList(Node* head){
    Node* temp = head;
    while (temp != nullptr){
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
```

```
int main(){
    Node* head = nullptr; // Initialize an empty list
```

```
// Add elements to the list  
addNodeAtMiddle(head, 3);  
addNodeAtMiddle(head, 1);  
addNodeAtMiddle(head, 4);  
addNodeAtMiddle(head, 5);  
addNodeAtMiddle(head, 2);  
  
// Display the list  
cout << "Linked list elements: ";  
displayList(head);  
  
return 0;  
}
```

OUTPUT

A screenshot of a code editor's console window. The window has a dark theme. At the top, there are tabs for 'Console' (active) and 'Shell'. Below the tabs, there are icons for a terminal, a refresh button, and a menu. The main area of the console shows the output of the program: 'Linked list elements: 1 4 2 5 3'. Above the output, there are buttons for 'Run', 'Ask AI', and a status bar showing '710ms • Just now' and a checkmark icon. The text 'Show Only Latest' and 'Clear History' are also visible in the top right of the console area.

```
>_ Console x Shell +  
Show Only Latest Clear History  
Run Ask AI 710ms • Just now ✓  
Linked list elements: 1 4 2 5 3
```

STACKS

1. Implement a stack using an array and perform push and pop operations.

CODE

```
#include <iostream>

using namespace std;

#define MAX 1000 // Define the maximum size of the stack

class Stack {

    int top;

    int arr[MAX]; // Array to store stack elements

public:

    Stack() { top = -1; } // Constructor to initialize top

    // Function to push an element onto the stack

    bool push(int x) {

        if (top >= (MAX - 1)) {

            cout << "Stack Overflow" << endl;

            return false;

        } else {
```

```
    arr[++top] = x;

    cout << x << " pushed onto stack" << endl;

    return true;

}

}
```

// Function to pop an element from the stack

```
int pop(){

    if (top < 0){

        cout << "Stack Underflow" << endl;

        return 0;

    } else {

        int x = arr[top--];

        return x;

    }

}
```

// Function to display the top element of the stack

```
int peek(){

    if (top < 0){

        cout << "Stack is Empty" << endl;

        return 0;

    }

}
```



```
    } else {  
        return arr[top];  
    }  
}
```

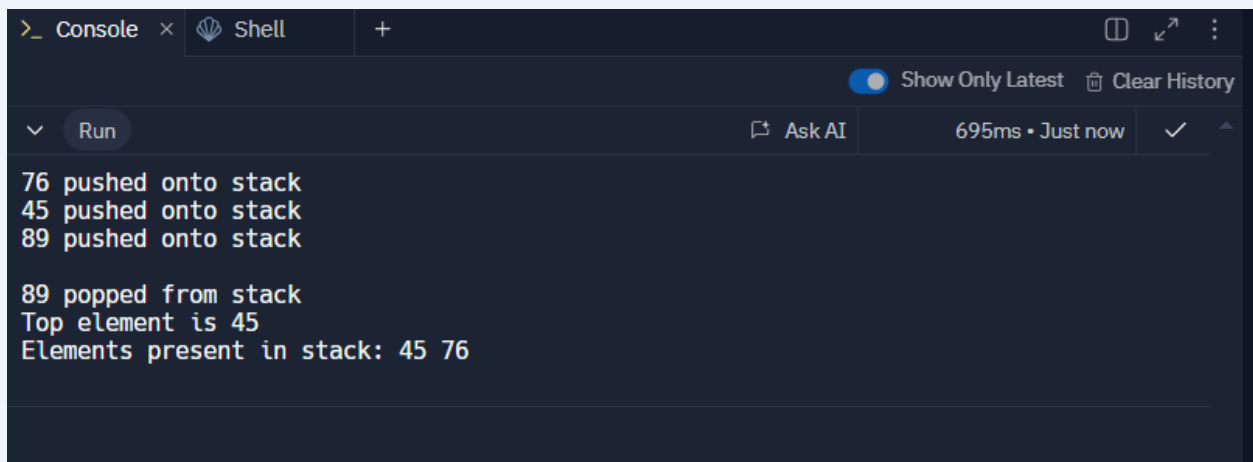
```
// Function to check if the stack is empty
```

```
bool isEmpty() {  
    return (top < 0);  
}  
};
```

```
int main() {  
    Stack stack;  
    stack.push(76);  
    stack.push(45);  
    stack.push(89);  
  
    cout << stack.pop() << " popped from stack" << endl;  
    cout << "Top element is " << stack.peek() << endl;  
  
    cout << "Elements present in stack: ";  
    while (!stack.isEmpty()) {
```

```
        cout << stack.pop() << " ";  
    }  
  
    return 0;  
}
```

OUTPUT



```
>_ Console x Shell +  
Show Only Latest Clear History  
Run Ask AI 695ms • Just now ✓  
76 pushed onto stack  
45 pushed onto stack  
89 pushed onto stack  
  
89 popped from stack  
Top element is 45  
Elements present in stack: 45 76
```

2. Write a program to check if a given string of parentheses is balanced (e.g., “(())” is balanced, but “(())” is not).

CODE

```
#include <iostream>  
  
#include <stack>  
  
#include <string>  
  
using namespace std;  
  
bool isBalanced(string expr)  
{
```

```

    stack<char> s;

    for (char ch : expr)
    {
        if (ch == '(')
        {
            s.push(ch);
        } else if (ch == ')')
        {
            if (s.empty() || s.top() != '(')
            {
                return false;
            }
            s.pop();
        }
    }

    return s.empty();
}

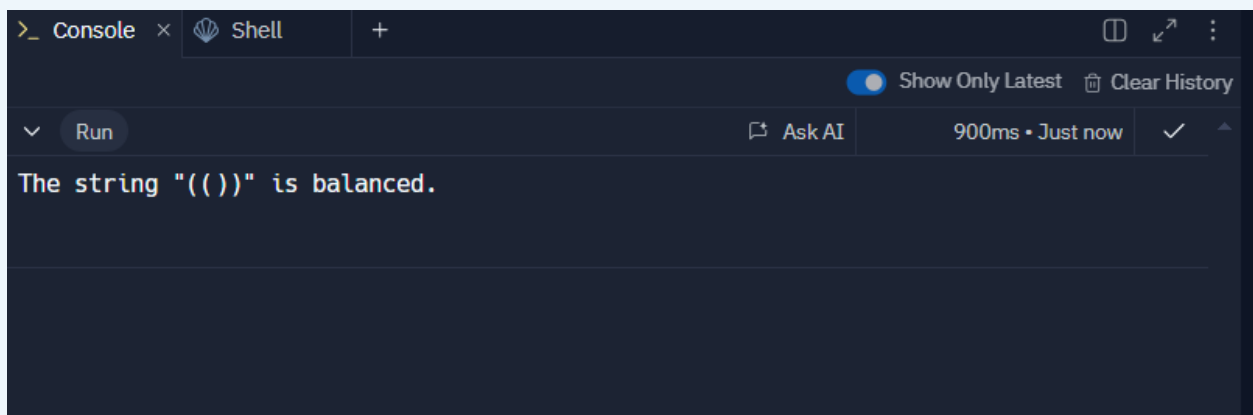
int main()
{
    string expr = "(() )"; // Example input

```

```
    if (isBalanced(expr))
    {
        cout << "The string \"" << expr << "\" is balanced." << endl;
    }
else
{
    cout << "The string \"" << expr << "\" is not balanced." << endl;
}

return 0;
}
```

OUTPUT

A screenshot of a code editor's console window. The window has a dark theme. At the top, there are tabs for 'Console' and 'Shell'. The 'Console' tab is active. Below the tabs, there is a toolbar with a 'Run' button, an 'Ask AI' button, and a status bar showing '900ms • Just now'. The main area of the console displays the output: 'The string "()" is balanced.'.

```
>_ Console x Shell +
Show Only Latest Clear History
Run Ask AI 900ms • Just now ✓
The string "()" is balanced.
```

3. Create a stack-based program to reverse a string (push each character and pop to reverse).

CODE

```
#include <iostream>

#include <stack>

#include <string>

using namespace std;

string reverseString(const string& str) {

    stack<char> s;

    for (char ch : str) {

        s.push(ch);

    }

    string reversed;

    while (!s.empty()) {

        reversed += s.top();

        s.pop();

    }

    return reversed;

}

int main() {

    string str = "hello"; // Example input
```

```
string reversedStr = reverseString(str);

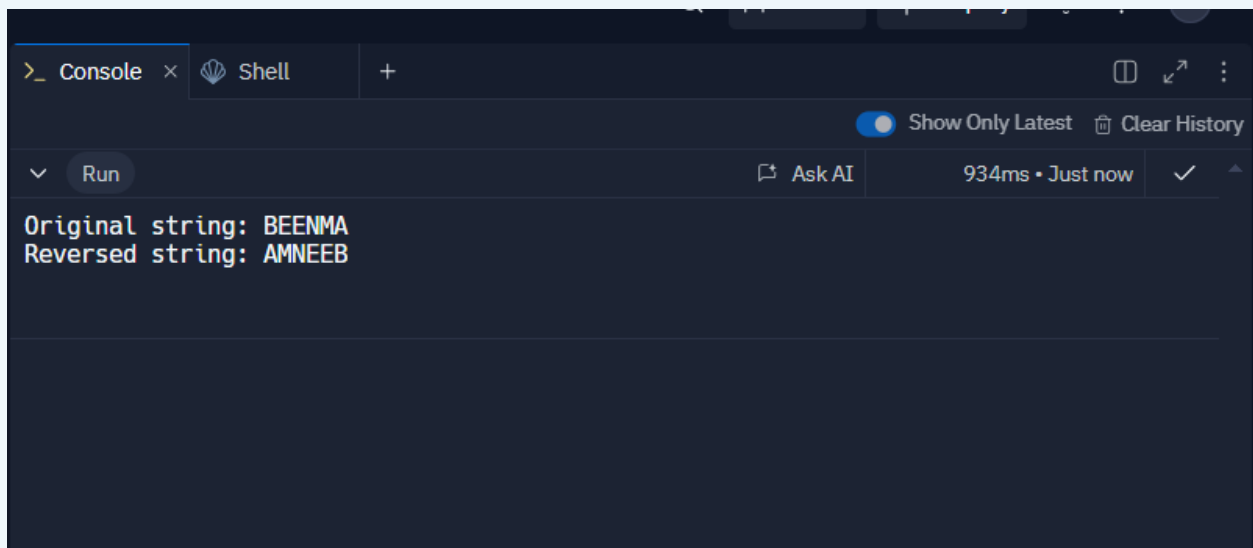
cout << "Original string: " << str << endl;

cout << "Reversed string: " << reversedStr << endl;

return 0;

}
```

OUTPUT

A screenshot of a code editor's console window. The window has a dark theme. At the top, there are tabs for 'Console' and 'Shell'. Below the tabs, there are buttons for 'Run', 'Ask AI', and a status bar showing '934ms • Just now'. The main area of the console displays the output of the program: 'Original string: BEENMA' and 'Reversed string: AMNEEB'.

QUEUES

1. Write a program to check if the elements in a queue form a palindrome. A palindrome reads the same forwards and backwards (e.g., '{1, 2, 3, 2, 1}').

CODE

```
#include <iostream>
```

```
#include <queue>
```

```
#include <stack>
```

```
#include <vector>
```

```
using namespace std;
```

```
bool isPalindrome(queue<int> q){
```

```
    stack<int> s;
```

```
    vector<int> original;
```

```
    while (!q.empty()){
```

```
        int value = q.front();
```

```
        q.pop();
```

```
        original.push_back(value);
```

```
        s.push(value);
```

```
    }
```

```
    for (int i = 0; i < original.size(); i++){
```

```
        if (original[i] != s.top()){
```

```
            return false;
```

```
        }
```

```
        s.pop();
```

```
}
```

```
return true;
```

```
}
```

```
int main() {
```

```
    queue<int> q;
```

```
    // Example input
```

```
    q.push(1);
```

```
    q.push(2);
```

```
    q.push(3);
```

```
    q.push(2);
```

```
    q.push(1);
```

```
    if (isPalindrome(q)) {
```

```
        cout << "The queue elements form a palindrome." << endl;
```

```
    } else {
```

```
        cout << "The queue elements do not form a palindrome." << endl;
```

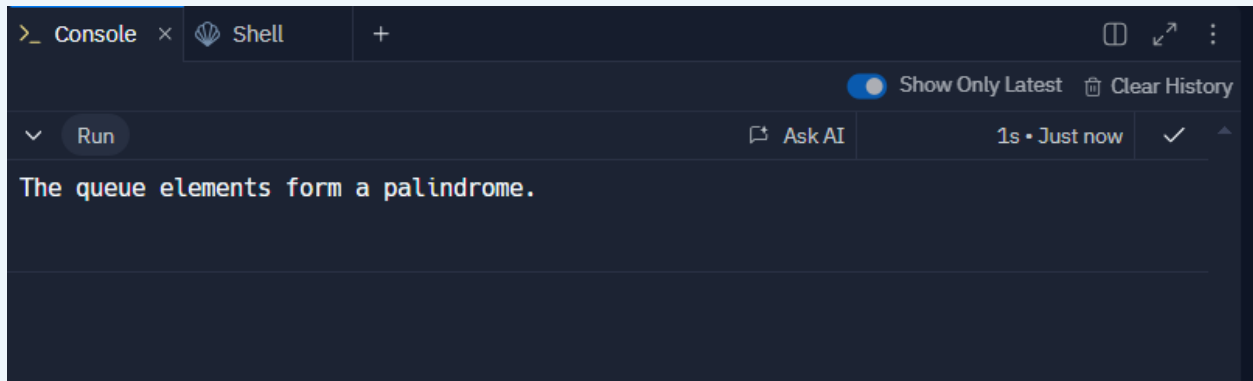
```
    }
```

```
    return 0;
```



```
}
```

OUTPUT

A screenshot of a code editor's console window. The window has tabs for 'Console' and 'Shell'. The 'Console' tab is active, showing a single line of output: 'The queue elements form a palindrome.' The console also displays a 'Run' button, an 'Ask AI' button, and a status bar indicating '1s • Just now'.

2. Implement a simple program that counts the total number of elements in a queue without modifying the queue's order.

CODE

```
#include <iostream>

#include <queue>

using namespace std;

int countQueueElements(queue<int> q) {

    int count = 0;

    while (!q.empty()) {

        count++;

        q.pop();

    }

}
```

```
        return count;
    }

int main() {
    queue<int> q;

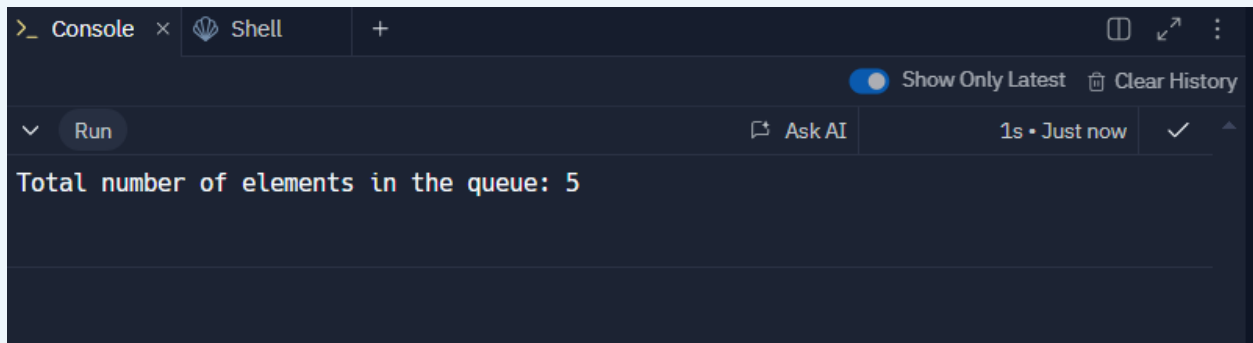
    // Example input
    q.push(1);
    q.push(2);
    q.push(3);
    q.push(4);
    q.push(5);

    int totalElements = countQueueElements(q);

    cout << "Total number of elements in the queue: " << totalElements << endl;

    return 0;
}
```

OUTPUT

A screenshot of a terminal window with a dark background. The top bar shows tabs for 'Console' and 'Shell', along with a '+' icon. On the right, there are icons for a window, a refresh arrow, and a menu. Below the tabs, a toggle switch is turned on, labeled 'Show Only Latest', followed by a trash icon and 'Clear History'. A 'Run' button is on the left, and 'Ask AI' is on the right. The main area displays the text 'Total number of elements in the queue: 5' in a monospaced font. At the bottom right, it shows '1s • Just now' and a checkmark icon.

```
>_ Console x Shell +  
Show Only Latest Clear History  
Run Ask AI 1s • Just now ✓  
Total number of elements in the queue: 5
```

3. Write a program to simulate a basic ticket queue, where people enter and leave the line in the order they joined.

CODE

```
#include <iostream>  
  
#include <queue>  
  
#include <string>  
  
using namespace std;  
  
int main() {  
    queue<string> ticketQueue;  
  
    // Simulate people entering the queue  
    ticketQueue.push("Alice");  
    ticketQueue.push("Bob");  
    ticketQueue.push("Charlie");  
    ticketQueue.push("Diana");
```

```
ticketQueue.push("Eve");

// Display the queue

cout << "People in the ticket queue: ";

queue<string> tempQueue = ticketQueue; // Copy queue to display without
modifying original

while (!tempQueue.empty()) {

    cout << tempQueue.front() << " ";

    tempQueue.pop();

}

cout << endl;


// Simulate people leaving the queue

while (!ticketQueue.empty()) {

    cout << ticketQueue.front() << " got the ticket and left the queue." << endl;

    ticketQueue.pop();

}


return 0;

}
```

OUTPUT

```
>_ Console x Shell +
Show Only Latest Clear History
Run Ask AI 1s • Just now ✓
People in the ticket queue:
Amna
Ayesha
Ahmed
Sana
Maryam

Amna got the ticket and left the queue.
Ayesha got the ticket and left the queue.
Ahmed got the ticket and left the queue.
Sana got the ticket and left the queue.
Maryam got the ticket and left the queue.
```

VECTORS

1. Write a program to add elements to a vector and display its size and capacity after each insertion.

CODE

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

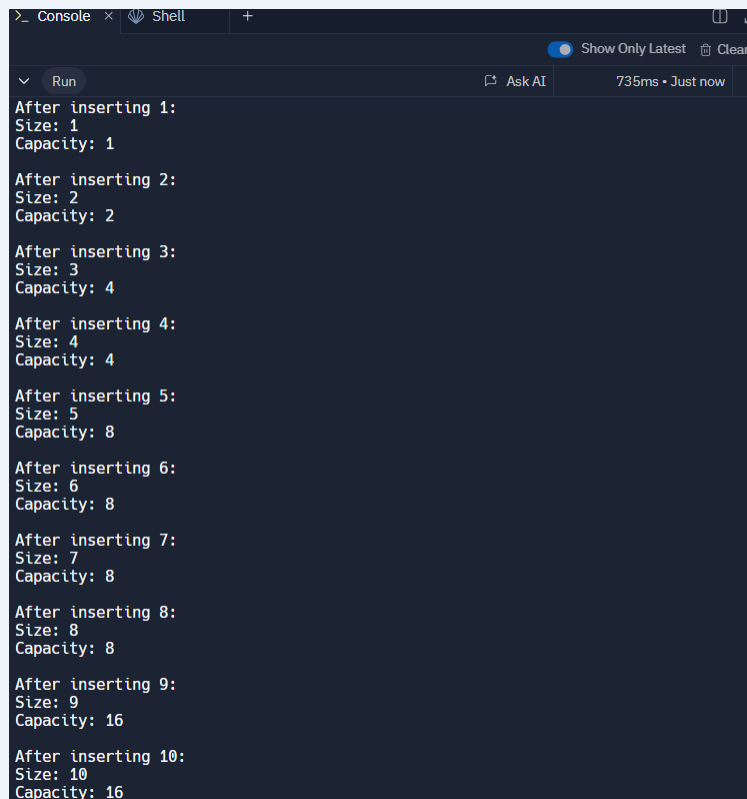
```
int main()
```

```
{
```

```
    vector<int> vec;
```

```
for (int i = 1; i <= 10; ++i){  
  
    vec.push_back(i);  
  
    cout << "After inserting " << i << ": " << endl;  
  
    cout << "Size: " << vec.size() << endl;  
  
    cout << "Capacity: " << vec.capacity() << endl << endl;  
  
}  
  
return 0;  
}
```

OUTPUT



```
> Console x Shell +  
Show Only Latest Clear  
Run Ask AI 735ms • Just now  
After inserting 1:  
Size: 1  
Capacity: 1  
  
After inserting 2:  
Size: 2  
Capacity: 2  
  
After inserting 3:  
Size: 3  
Capacity: 4  
  
After inserting 4:  
Size: 4  
Capacity: 4  
  
After inserting 5:  
Size: 5  
Capacity: 8  
  
After inserting 6:  
Size: 6  
Capacity: 8  
  
After inserting 7:  
Size: 7  
Capacity: 8  
  
After inserting 8:  
Size: 8  
Capacity: 8  
  
After inserting 9:  
Size: 9  
Capacity: 16  
  
After inserting 10:  
Size: 10  
Capacity: 16
```

2. Implement a program that removes duplicate values from a vector.

CODE

```
#include <iostream>

#include <vector>

#include <algorithm>

#include <unordered_set>

using namespace std;

void removeDuplicates(vector<int>& vec) {
    unordered_set<int> seen;
    auto end = remove_if(vec.begin(), vec.end(), [&seen](int value) {
        if (seen.find(value) != seen.end()) {
            return true;
        } else {
            seen.insert(value);
            return false;
        }
    });
    vec.erase(end, vec.end());
}
```

```
int main()
{
    vector<int> vec = {1, 2, 2, 3, 4, 4, 5, 1, 6, 7};

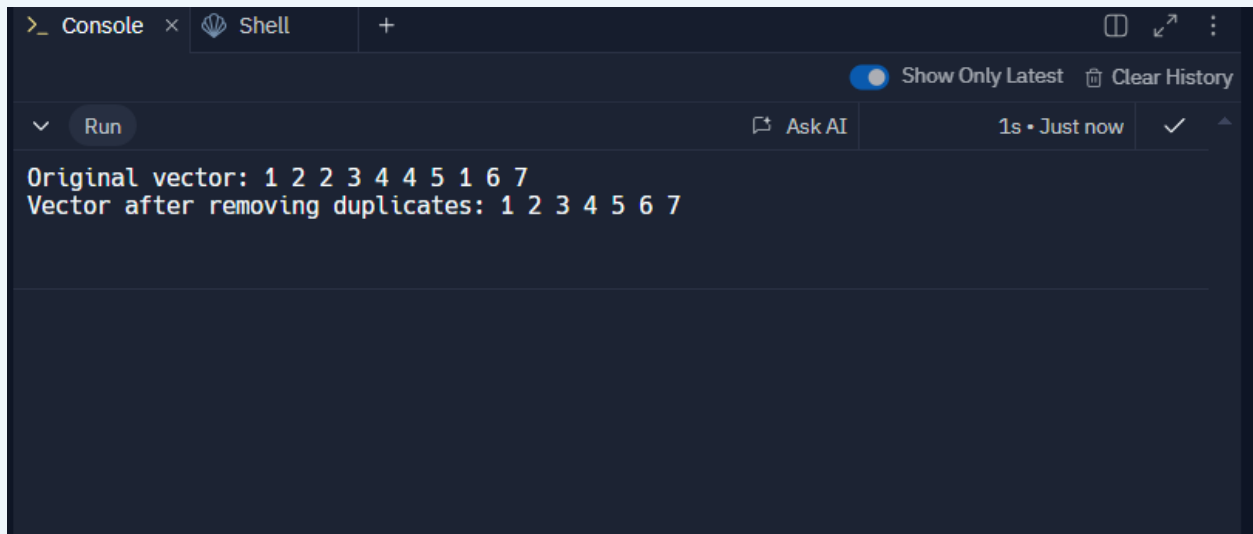
    cout << "Original vector: ";
    for (int val : vec) {
        cout << val << " ";
    }
    cout << endl;

    removeDuplicates(vec);

    cout << "Vector after removing duplicates: ";
    for (int val : vec) {
        cout << val << " ";
    }
    cout << endl;

    return 0;
}
```

OUTPUT

A screenshot of a code editor's console window. The window has tabs for 'Console' and 'Shell'. The console output shows two lines: 'Original vector: 1 2 2 3 4 4 5 1 6 7' and 'Vector after removing duplicates: 1 2 3 4 5 6 7'. The interface includes a 'Run' button, an 'Ask AI' button, and a 'Show Only Latest' toggle. The execution time is shown as '1s • Just now'.

3. Create a program to sort a vector of integers in ascending order.

CODE

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main(){

    vector<int> vec = {5, 1, 4, 2, 8}; // Example vector

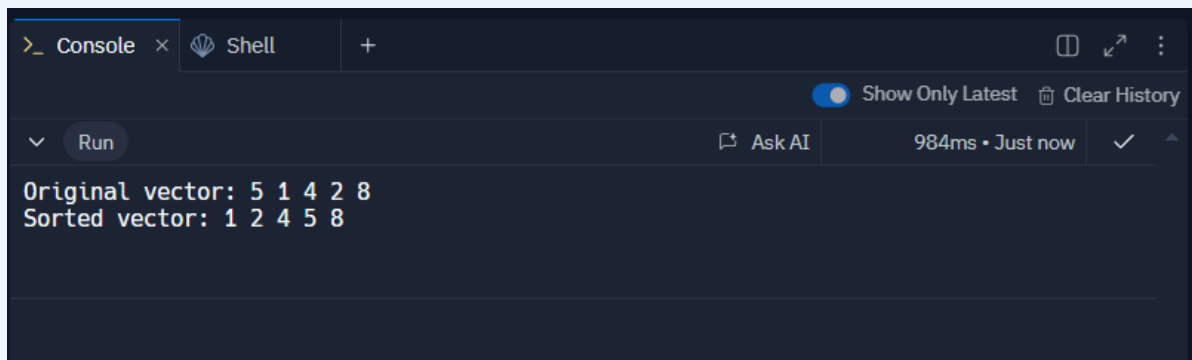
    cout << "Original vector: ";

    for (int val : vec){

        cout << val << " ";
```

```
}  
  
cout << endl;  
  
sort(vec.begin(), vec.end()); // Sorting the vector in ascending order  
  
cout << "Sorted vector: ";  
for (int val : vec) {  
    cout << val << " ";  
}  
  
cout << endl;  
  
return 0;  
}
```

OUTPUT



The screenshot shows a dark-themed IDE window with a 'Console' tab active. The console output displays the original and sorted vectors. The 'Run' button is visible, along with an 'Ask AI' feature and a status bar indicating a 984ms execution time.

```
>_ Console x Shell +  
Show Only Latest Clear History  
Run Ask AI 984ms • Just now ✓  
Original vector: 5 1 4 2 8  
Sorted vector: 1 2 4 5 8
```