

Name – Adya Singh

Reg No – RA2211003010181

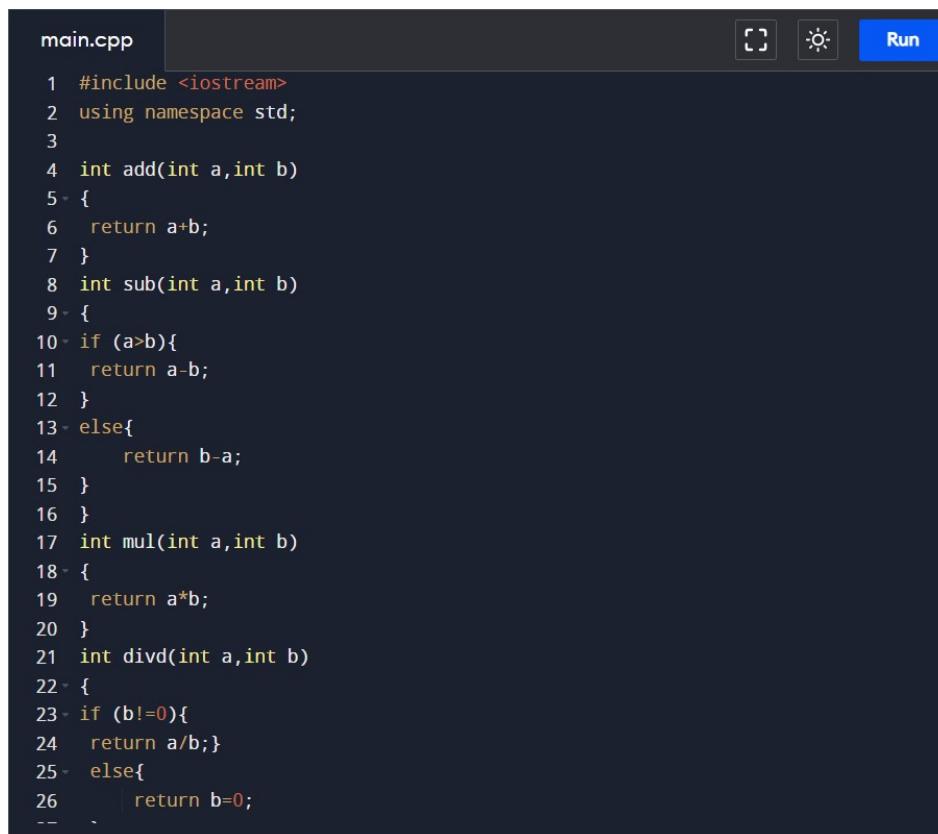
21CSC203P

Advanced Programming Practice

Assignment 3

1. Simple mini calculator program in C++ that uses subroutines for basic arithmetic operations.

Code –



```
main.cpp
1 #include <iostream>
2 using namespace std;
3
4 int add(int a,int b)
5 {
6     return a+b;
7 }
8 int sub(int a,int b)
9 {
10    if (a>b){
11        return a-b;
12    }
13    else{
14        return b-a;
15    }
16 }
17 int mul(int a,int b)
18 {
19     return a*b;
20 }
21 int divd(int a,int b)
22 {
23    if (b!=0){
24        return a/b; }
25    else{
26        return b=0;
27    }
}
```

main.cpp

```
26     | return b=0;
27 }
28 }
29 int main() {
30     int a,b;
31     cout<<"Enter the number 1:" ;
32     cin>>a;
33     cout<<"Enter the number 2:" ;
34     cin>>b;
35     char op;
36     cout<<"Enter operation to done:" ;
37     cin>>op;
38     int result;
39     switch(op){
40         case '+':
41             result=add(a,b);
42             break;
43         case '-':
44             result=sub(a,b);
45             break;
46         case '*':
47             result=mul(a,b);
48             break;
49         case '/':
50             result=divd(a,b);
51             break;
```

main.cpp

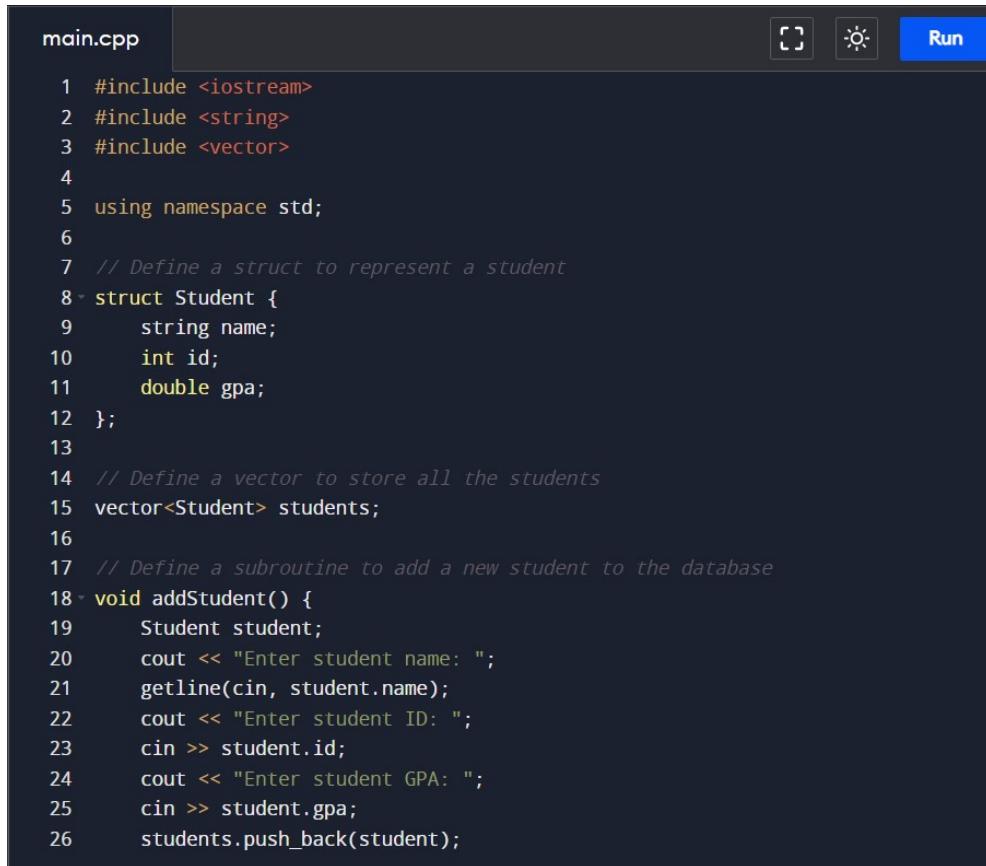
```
33     cout<<"Enter the number 2:" ;
34     cin>>b;
35     char op;
36     cout<<"Enter operation to done:" ;
37     cin>>op;
38     int result;
39     switch(op){
40         case '+':
41             result=add(a,b);
42             break;
43         case '-':
44             result=sub(a,b);
45             break;
46         case '*':
47             result=mul(a,b);
48             break;
49         case '/':
50             result=divd(a,b);
51             break;
52         default:
53             cout<<"Invalid operator";
54             return 1;
55     }
56     cout<<"The answer is :"<<result;
57     return 0;
58 }
```

Output –

```
Output Clear
/tmp/qmdKgu9F71.o
Enter the number 1:500
Enter the number 2:100
Enter operation to done:/
The answer is :5|
```

2. Write a complete students database with subroutines involves storing and managing student information using appropriate data structures and providing various functionalities to interact with the database and implement in C++ with subroutines:

Code –



The screenshot shows a code editor window with a dark theme. The file tab at the top left is labeled "main.cpp". To the right of the tab are three icons: a square with two overlapping circles, a sun-like icon, and a blue "Run" button. The code area contains the following C++ code:

```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4
5 using namespace std;
6
7 // Define a struct to represent a student
8 struct Student {
9     string name;
10    int id;
11    double gpa;
12 };
13
14 // Define a vector to store all the students
15 vector<Student> students;
16
17 // Define a subroutine to add a new student to the database
18 void addStudent() {
19     Student student;
20     cout << "Enter student name: ";
21     getline(cin, student.name);
22     cout << "Enter student ID: ";
23     cin >> student.id;
24     cout << "Enter student GPA: ";
25     cin >> student.gpa;
26     students.push_back(student);
```

main.cpp



```
26     students.push_back(student);
27     cout << "Student added successfully!" << endl;
28 }
29
30 // Define a subroutine to print all the students in the database
31 void printStudents() {
32     if (students.empty()) {
33         cout << "No students in the database." << endl;
34     } else {
35         cout << "Students in the database:" << endl;
36         for (const auto& student : students) {
37             cout << "Name: " << student.name << ", ID: " << student.id << ", "
38             GPA: " << student.gpa << endl;
39         }
40     }
41
42 // Define a subroutine to search for a student by ID
43 void searchStudent() {
44     int id;
45     cout << "Enter student ID to search for: ";
46     cin >> id;
47     for (const auto& student : students) {
48         if (student.id == id) {
49             cout << "Student found:" << endl;
50             cout << "Name: " << student.name << ", ID: " << student.id << ", "
51             GPA: " << student.gpa << endl;
52         }
53     }
54     cout << "Student not found." << endl;
55 }
56
57 // Define a subroutine to delete a student by ID
58 void deleteStudent() {
59     int id;
60     cout << "Enter student ID to delete: ";
61     cin >> id;
62     for (auto it = students.begin(); it != students.end(); ++it) {
63         if (it->id == id) {
64             students.erase(it);
65             cout << "Student deleted successfully!" << endl;
66             return;
67         }
68     }
69     cout << "Student not found." << endl;
70 }
71
72 // Define the main function to interact with the user
73 int main() {
74     while (true) {
75         cout << "Select an option:" << endl;
```

main.cpp



```
51         return;
52     }
53 }
54 cout << "Student not found." << endl;
55 }
56
57 // Define a subroutine to delete a student by ID
58 void deleteStudent() {
59     int id;
60     cout << "Enter student ID to delete: ";
61     cin >> id;
62     for (auto it = students.begin(); it != students.end(); ++it) {
63         if (it->id == id) {
64             students.erase(it);
65             cout << "Student deleted successfully!" << endl;
66             return;
67         }
68     }
69     cout << "Student not found." << endl;
70 }
71
72 // Define the main function to interact with the user
73 int main() {
74     while (true) {
75         cout << "Select an option:" << endl;
```

main.cpp

```
75     cout << "Select an option:" << endl;
76     cout << "1. Add a student" << endl;
77     cout << "2. Print all students" << endl;
78     cout << "3. Search for a student" << endl;
79     cout << "4. Delete a student" << endl;
80     cout << "5. Quit" << endl;
81     int option;
82     cin >> option;
83     cin.ignore(); // Ignore the newline character left by cin
84     switch (option) {
85         case 1:
86             addStudent();
87             break;
88         case 2:
89             printStudents();
90             break;
91         case 3:
92             searchStudent();
93             break;
94         case 4:
95             deleteStudent();
96             break;
97         case 5:
98             return 0;
99         default:
100             cout << "Invalid option. Please try again." << endl;
```

main.cpp

```
79     cout << "4. Delete a student" << endl;
80     cout << "5. Quit" << endl;
81     int option;
82     cin >> option;
83     cin.ignore(); // Ignore the newline character left by cin
84     switch (option) {
85         case 1:
86             addStudent();
87             break;
88         case 2:
89             printStudents();
90             break;
91         case 3:
92             searchStudent();
93             break;
94         case 4:
95             deleteStudent();
96             break;
97         case 5:
98             return 0;
99         default:
100             cout << "Invalid option. Please try again." << endl;
101     }
102 }
103 }
```

Output –

```
Output Clear
/tmp/SMInIj0fRD.o
Select an option:
1. Add a student
2. Print all students
3. Search for a student
4. Delete a student
5. Quit
1
Enter student name: Adya Singh
Enter student ID: 181
Enter student GPA: 9.8
Student added successfully!
Select an option:
1. Add a student
2. Print all students
3. Search for a student
4. Delete a student
5. Quit
2
Students in the database:
Name: Adya Singh, ID: 181, GPA: 9.8
Select an option:
1. Add a student
2. Print all students
3. Search for a student
4. Delete a student
```

```
Output Clear
Select an option:
1. Add a student
2. Print all students
3. Search for a student
4. Delete a student
5. Quit
3
Enter student ID to search for: 181
Student found:
Name: Adya Singh, ID: 181, GPA: 9.8
Select an option:
1. Add a student
2. Print all students
3. Search for a student
4. Delete a student
5. Quit
4
Enter student ID to delete: 181
Student deleted successfully!
Select an option:
1. Add a student
2. Print all students
3. Search for a student
4. Delete a student
5. Quit
```

3. Design a subroutine program to calculate the area and perimeter of different geometric shapes (circle, rectangle, triangle, etc.).

Code –

The screenshot shows a code editor window with the file name "main.cpp" at the top left. At the top right, there are three icons: a square with two overlapping circles, a sun-like icon, and a blue "Run" button. The code itself is written in C++ and defines three functions to calculate the properties of a circle, a rectangle, and a triangle.

```
1 #include <iostream>
2 #include <cmath>
3
4 // Function to calculate the area and perimeter of a circle
5 void circleProperties(double radius, double& area, double& perimeter) {
6     const double pi = 3.14159;
7     area = pi * radius * radius;
8     perimeter = 2 * pi * radius;
9 }
10
11 // Function to calculate the area and perimeter of a rectangle
12 void rectangleProperties(double length, double width, double& area, double&
13     perimeter) {
14     area = length * width;
15     perimeter = 2 * (length + width);
16 }
17 // Function to calculate the area and perimeter of a triangle
18 void triangleProperties(double side1, double side2, double side3, double& area,
19     double& perimeter) {
20     double s = (side1 + side2 + side3) / 2;
21     area = std::sqrt(s * (s - side1) * (s - side2) * (s - side3));
22     perimeter = side1 + side2 + side3;
23 }
24 int main() {
```

```
main.cpp
```

```
24 int main() {
25     int choice;
26     double area, perimeter;
27
28     do {
29         std::cout << "Select a shape to calculate properties:\n";
30         std::cout << "1. Circle\n";
31         std::cout << "2. Rectangle\n";
32         std::cout << "3. Triangle\n";
33         std::cout << "4. Exit\n";
34         std::cout << "Enter your choice: ";
35         std::cin >> choice;
36
37         switch (choice) {
38             case 1: {
39                 double radius;
40                 std::cout << "Enter the radius of the circle: ";
41                 std::cin >> radius;
42                 circleProperties(radius, area, perimeter);
43                 std::cout << "Area: " << area << "\n";
44                 std::cout << "Perimeter: " << perimeter << "\n";
45                 break;
46             }
47             case 2: {
48                 double length, width;
49                 std::cout << "Enter the length of the rectangle: ";
```

```
main.cpp
```

```
50                     std::cout << "Enter the length of the rectangle: ";
51                     std::cin >> length;
52                     std::cout << "Enter the width of the rectangle: ";
53                     std::cin >> width;
54                     rectangleProperties(length, width, area, perimeter);
55                     std::cout << "Area: " << area << "\n";
56                     std::cout << "Perimeter: " << perimeter << "\n";
57                     break;
58             }
59             case 3: {
60                 double side1, side2, side3;
61                 std::cout << "Enter the length of side 1 of the triangle: ";
62                 std::cin >> side1;
63                 std::cout << "Enter the length of side 2 of the triangle: ";
64                 std::cin >> side2;
65                 std::cout << "Enter the length of side 3 of the triangle: ";
66                 std::cin >> side3;
67                 triangleProperties(side1, side2, side3, area, perimeter);
68                 std::cout << "Area: " << area << "\n";
69                 std::cout << "Perimeter: " << perimeter << "\n";
70                 break;
71             }
72             case 4:
73                 std::cout << "Exiting the program.\n";
74                 break;
75             default:
```

```
main.cpp

58     case 3: {
59         double side1, side2, side3;
60         std::cout << "Enter the length of side 1 of the triangle: ";
61         std::cin >> side1;
62         std::cout << "Enter the length of side 2 of the triangle: ";
63         std::cin >> side2;
64         std::cout << "Enter the length of side 3 of the triangle: ";
65         std::cin >> side3;
66         triangleProperties(side1, side2, side3, area, perimeter);
67         std::cout << "Area: " << area << "\n";
68         std::cout << "Perimeter: " << perimeter << "\n";
69         break;
70     }
71     case 4:
72         std::cout << "Exiting the program.\n";
73         break;
74     default:
75         std::cout << "Invalid choice. Try again.\n";
76     }
77     std::cout << std::endl;
78 } while (choice != 4);
79
80 return 0;
81 }
82
83 |
```

Output –

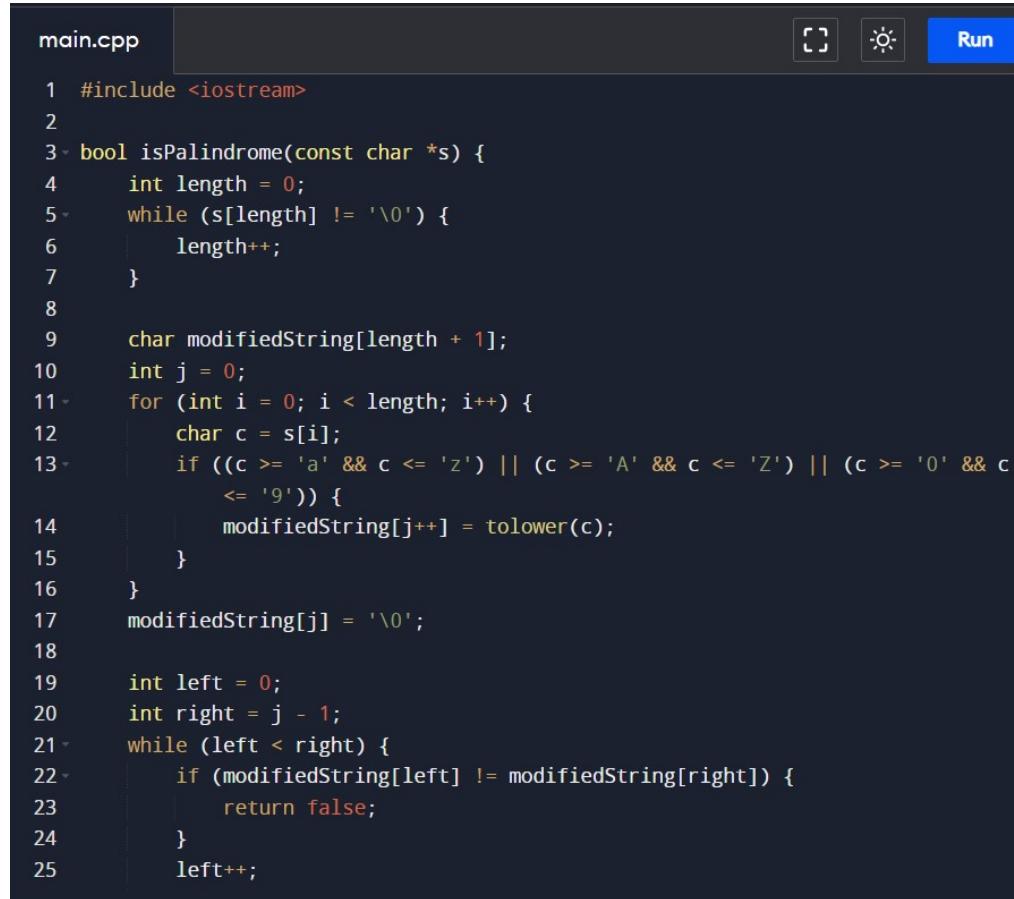
```
Output
Clear

/tmp/Zsz9THHMvC.o
Select a shape to calculate properties:
1. Circle
2. Rectangle
3. Triangle
4. Exit
Enter your choice: 1
Enter the radius of the circle: 6.5
Area: 132.732
Perimeter: 40.8407

Select a shape to calculate properties:
1. Circle
2. Rectangle
3. Triangle
4. Exit
Enter your choice: 4
Exiting the program.
```

4. Implement a subroutine program to check if a given string is a palindrome or not.

Code –



```
main.cpp

1 #include <iostream>
2
3 bool isPalindrome(const char *s) {
4     int length = 0;
5     while (s[length] != '\0') {
6         length++;
7     }
8
9     char modifiedString[length + 1];
10    int j = 0;
11    for (int i = 0; i < length; i++) {
12        char c = s[i];
13        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c
14            <= '9')) {
15            modifiedString[j++] = tolower(c);
16        }
17    }
18    modifiedString[j] = '\0';
19    int left = 0;
20    int right = j - 1;
21    while (left < right) {
22        if (modifiedString[left] != modifiedString[right]) {
23            return false;
24        }
25        left++;
26    }
27    return true;
28}
```

```
main.cpp
```

The screenshot shows a code editor window with the file name "main.cpp" at the top left. The code itself is as follows:

```
18     int left = 0;
19     int right = j - 1;
20     while (left < right) {
21         if (modifiedString[left] != modifiedString[right]) {
22             return false;
23         }
24         left++;
25         right--;
26     }
27 }
28 return true;
29 }
30
31 int main() {
32     char input_string[100];
33     std::cout << "Enter a string: ";
34     std::cin.getline(input_string, sizeof(input_string));
35
36     if (isPalindrome(input_string)) {
37         std::cout << "The given string is a palindrome." << std::endl;
38     } else {
39         std::cout << "The given string is not a palindrome." << std::endl;
40     }
41
42     return 0;
43 }
```

Output –

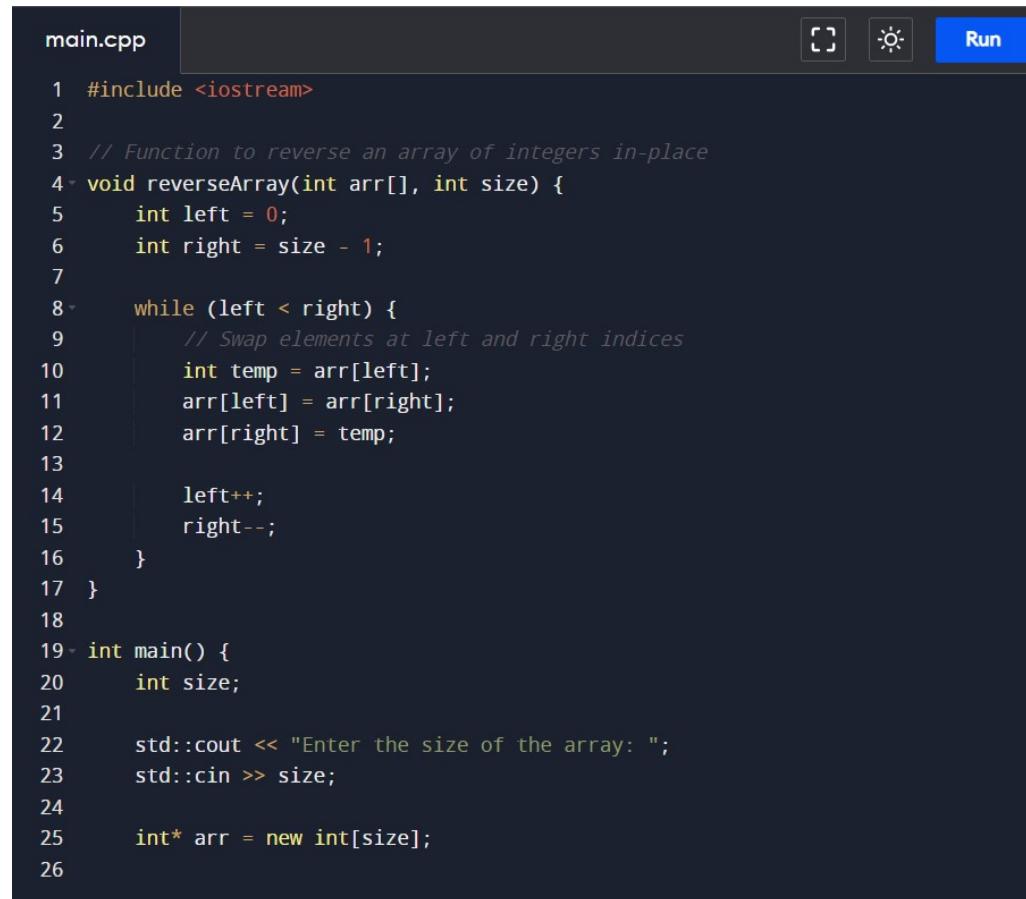
```
Output
```

The screenshot shows a terminal window with the title "Output" at the top left. At the top right is a "Clear" button. The terminal displays the following text:

```
/tmp/EYbygeXR2s.o
Enter a string: level
The given string is a palindrome.
```

5. Implement a subroutine program to reverse an array of integers in-place.

Code –



The screenshot shows a code editor window with the following details:

- Title Bar:** The title bar displays "main.cpp".
- Toolbar:** A toolbar at the top right contains three icons: a square with two vertical bars, a sun-like icon, and a blue "Run" button.
- Code Area:** The main area contains 26 lines of C++ code. Lines 1 through 18 are part of the reverseArray function, while lines 19 through 26 are part of the main function.
- Code Content:**

```
1 #include <iostream>
2
3 // Function to reverse an array of integers in-place
4 void reverseArray(int arr[], int size) {
5     int left = 0;
6     int right = size - 1;
7
8     while (left < right) {
9         // Swap elements at left and right indices
10        int temp = arr[left];
11        arr[left] = arr[right];
12        arr[right] = temp;
13
14        left++;
15        right--;
16    }
17 }
18
19 int main() {
20     int size;
21
22     std::cout << "Enter the size of the array: ";
23     std::cin >> size;
24
25     int* arr = new int[size];
26 }
```

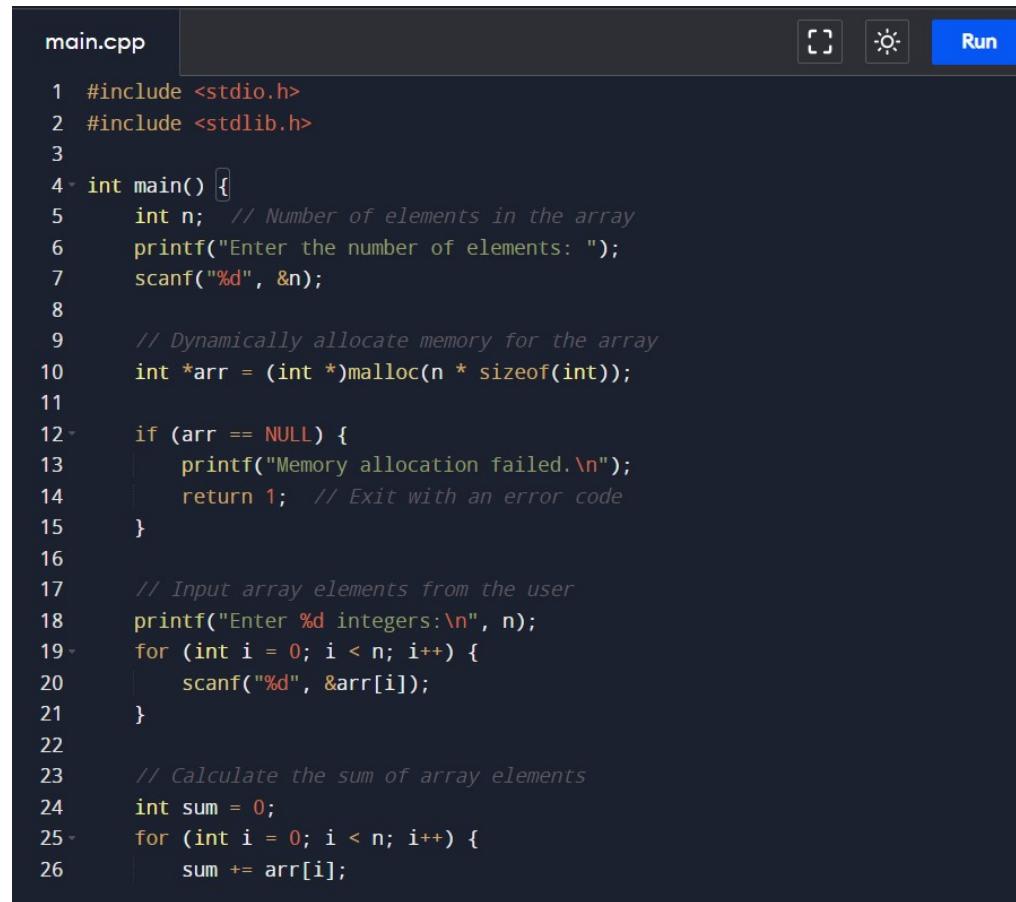
```
main.cpp | Run |     
~.  
25     int* arr = new int[size];  
26  
27     std::cout << "Enter " << size << " integers: ";  
28+    for (int i = 0; i < size; i++) {  
29         std::cin >> arr[i];  
30     }  
31  
32     std::cout << "Original array: ";  
33+    for (int i = 0; i < size; i++) {  
34         std::cout << arr[i] << " ";  
35     }  
36     std::cout << std::endl;  
37  
38     reverseArray(arr, size);  
39  
40     std::cout << "Reversed array: ";  
41+    for (int i = 0; i < size; i++) {  
42         std::cout << arr[i] << " ";  
43     }  
44     std::cout << std::endl;  
45  
46     delete[] arr;  
47  
48     return 0;  
49 }  
50
```

Output –

```
Output | Clear |     
/tmp/siYagOqKiM.o  
Enter the size of the array: 10  
Enter 10 integers: 1 2 3 4 5 6 7 8 9 10  
Original array: 1 2 3 4 5 6 7 8 9 10  
Reversed array: 10 9 8 7 6 5 4 3 2 1
```

6. Write a program that dynamically allocates memory for an array of integers based on user input and then finds the sum of all elements in the array.

Code –



The screenshot shows a code editor window titled "main.cpp". The code is a C++ program that performs the following tasks:

- Includes `<stdio.h>` and `<stdlib.h>`.
- Defines the `main()` function.
- Asks the user for the number of elements in the array (`n`).
- Dynamically allocates memory for the array (`*arr`) using `malloc(n * sizeof(int))`.
- If memory allocation fails, it prints an error message and exits with code 1.
- Inputs array elements from the user.
- Calculates the sum of array elements.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int n; // Number of elements in the array
6     printf("Enter the number of elements: ");
7     scanf("%d", &n);
8
9     // Dynamically allocate memory for the array
10    int *arr = (int *)malloc(n * sizeof(int));
11
12    if (arr == NULL) {
13        printf("Memory allocation failed.\n");
14        return 1; // Exit with an error code
15    }
16
17    // Input array elements from the user
18    printf("Enter %d integers:\n", n);
19    for (int i = 0; i < n; i++) {
20        scanf("%d", &arr[i]);
21    }
22
23    // Calculate the sum of array elements
24    int sum = 0;
25    for (int i = 0; i < n; i++) {
26        sum += arr[i];
```

```
main.cpp
```

```
10     int *arr = (int *)malloc(n * sizeof(int));
11
12     if (arr == NULL) {
13         printf("Memory allocation failed.\n");
14         return 1; // Exit with an error code
15     }
16
17     // Input array elements from the user
18     printf("Enter %d integers:\n", n);
19     for (int i = 0; i < n; i++) {
20         scanf("%d", &arr[i]);
21     }
22
23     // Calculate the sum of array elements
24     int sum = 0;
25     for (int i = 0; i < n; i++) {
26         sum += arr[i];
27     }
28
29     printf("Sum of array elements: %d\n", sum);
30
31     // Free the dynamically allocated memory
32     free(arr);
33
34     return 0; // Exit successfully
35 }
```

Output –

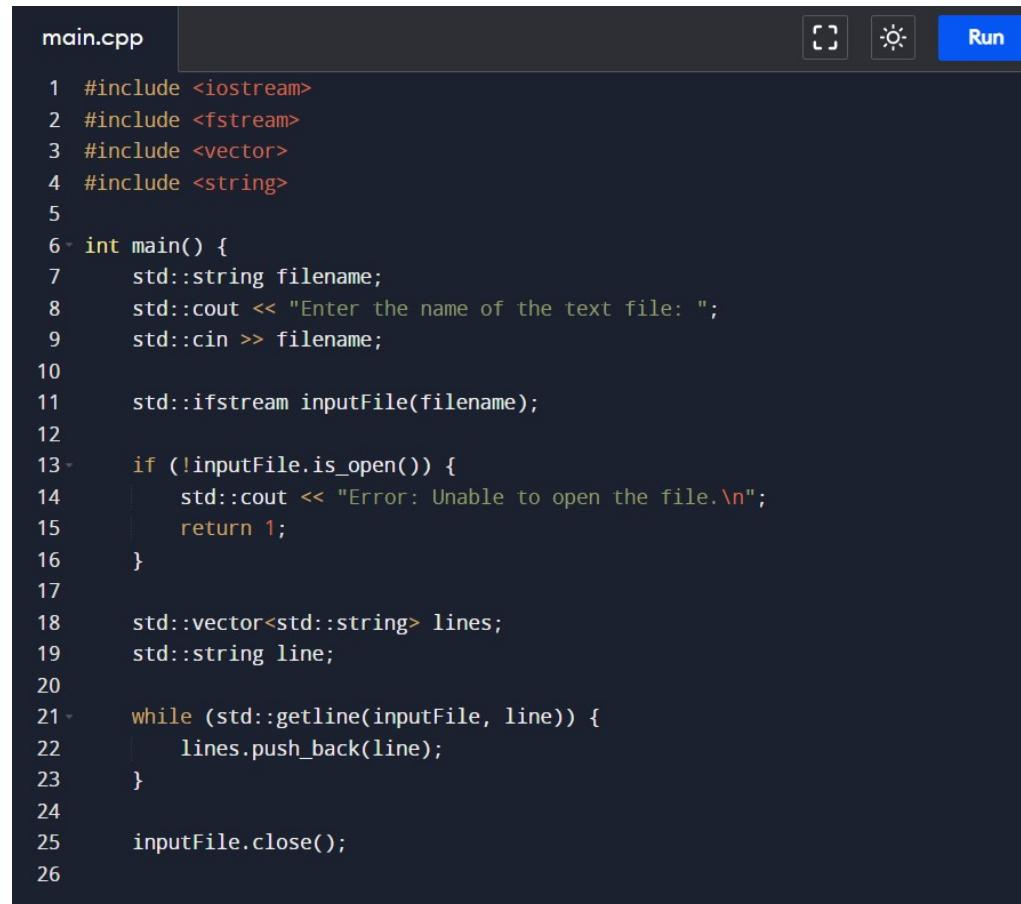
Output

Clear

```
/tmp/MxIDdR0kLc.o
Enter the number of elements: 5
Enter 5 integers:
4 7 44 76 55
Sum of array elements: 186
```

7. Implement a program that reads a text file and dynamically stores each line as a string in memory. Then, display the content of the file with line numbers.

Code –



The screenshot shows a code editor window titled "main.cpp". The code is a C++ program that reads a text file and stores its lines in a vector of strings. The code includes file I/O, exception handling, and a vector container. The code editor has a dark theme with syntax highlighting for C++ keywords and comments. The status bar at the bottom shows the file name "main.cpp" and the line number "1".

```
1 #include <iostream>
2 #include <fstream>
3 #include <vector>
4 #include <string>
5
6 int main() {
7     std::string filename;
8     std::cout << "Enter the name of the text file: ";
9     std::cin >> filename;
10
11    std::ifstream inputFile(filename);
12
13    if (!inputFile.is_open()) {
14        std::cout << "Error: Unable to open the file.\n";
15        return 1;
16    }
17
18    std::vector<std::string> lines;
19    std::string line;
20
21    while (std::getline(inputFile, line)) {
22        lines.push_back(line);
23    }
24
25    inputFile.close();
26}
```

```
main.cpp | [ ] Run

14     std::cout << "Error: Unable to open the file.\n";
15     return 1;
16 }
17
18 std::vector<std::string> lines;
19 std::string line;
20
21 while (std::getline(inputFile, line)) {
22     lines.push_back(line);
23 }
24
25 inputFile.close();
26
27 if (lines.empty()) {
28     std::cout << "The file is empty.\n";
29     return 0;
30 }
31
32 std::cout << "Content of the file with line numbers:\n";
33 for (size_t i = 0; i < lines.size(); i++) {
34     std::cout << i + 1 << ". " << lines[i] << std::endl;
35 }
36
37 return 0;
38 }
39
```

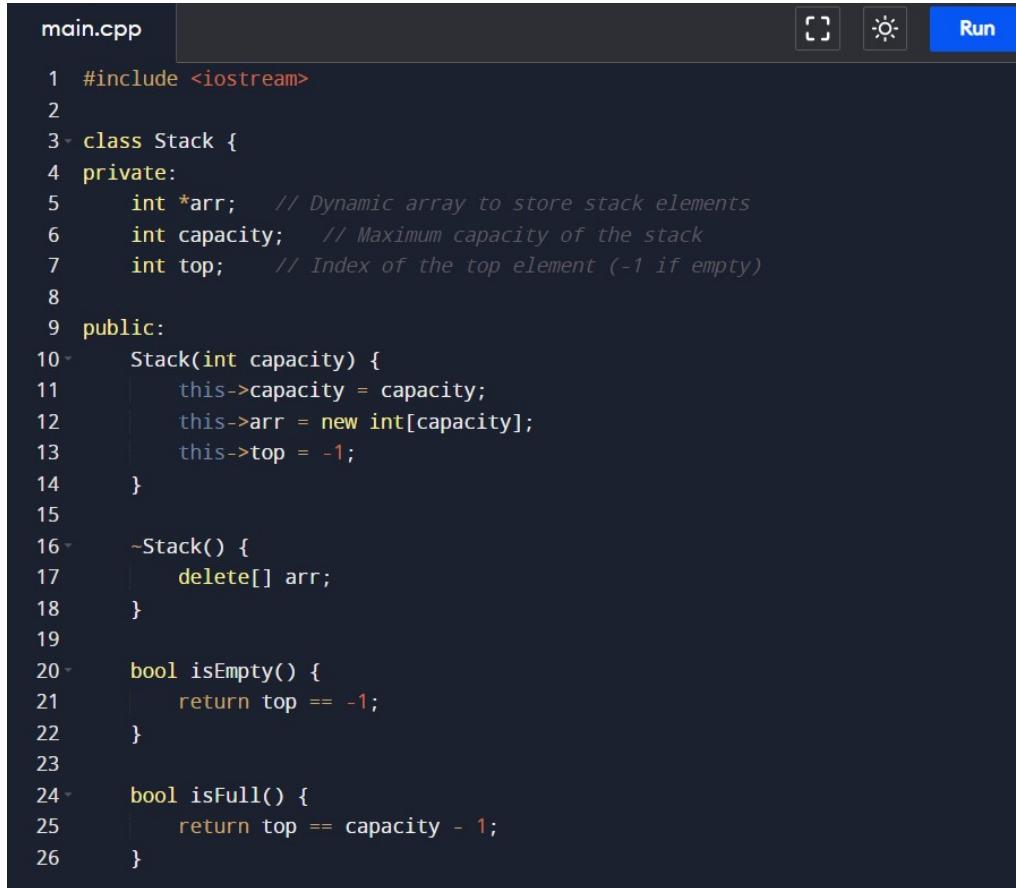
Output –

```
Output | Clear

/tmp/XyDEBcgUE.o
Enter the name of the text file: addie.txt
ERROR!
Error: Unable to open the file.
```

8. Create a program that uses dynamic memory allocation to implement a stack data structure to push and pop elements.

Code –



The screenshot shows a code editor window with the file name "main.cpp" at the top left. At the top right, there are three icons: a square with two vertical bars, a sun-like icon, and a blue "Run" button. The main area contains the following C++ code:

```
1 #include <iostream>
2
3 class Stack {
4 private:
5     int *arr; // Dynamic array to store stack elements
6     int capacity; // Maximum capacity of the stack
7     int top; // Index of the top element (-1 if empty)
8
9 public:
10 Stack(int capacity) {
11     this->capacity = capacity;
12     this->arr = new int[capacity];
13     this->top = -1;
14 }
15
16 ~Stack() {
17     delete[] arr;
18 }
19
20 bool isEmpty() {
21     return top == -1;
22 }
23
24 bool isFull() {
25     return top == capacity - 1;
26 }
```

```
main.cpp | Run
```

```
26     }
27
28+ void push(int value) {
29     if (isFull()) {
30         std::cout << "Stack is full. Cannot push element.\n";
31         return;
32     }
33     top++;
34     arr[top] = value;
35 }
36
37+ void pop() {
38     if (isEmpty()) {
39         std::cout << "Stack is empty. Cannot pop element.\n";
40         return;
41     }
42     top--;
43 }
44
45+ int peek() {
46     if (isEmpty()) {
47         std::cout << "Stack is empty. Cannot peek.\n";
48         return -1;
49     }
50     return arr[top];
51 }
```

```
main.cpp | Run
```

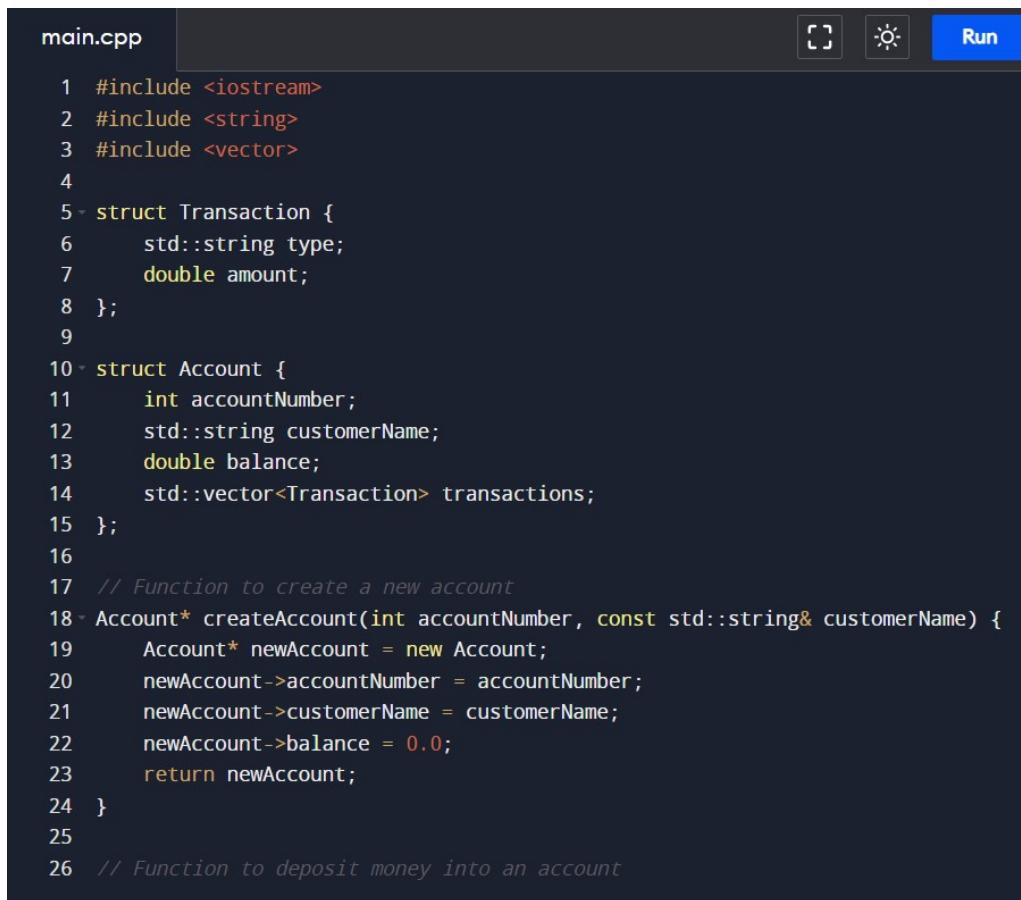
```
48         return -1;
49     }
50     return arr[top];
51 }
52 };
53
54+ int main() {
55     int capacity;
56     std::cout << "Enter the capacity of the stack: ";
57     std::cin >> capacity;
58
59     Stack stack(capacity);
60
61     stack.push(10);
62     stack.push(20);
63     stack.push(30);
64
65     std::cout << "Top element: " << stack.peek() << std::endl;
66
67     stack.pop();
68
69     std::cout << "Top element after pop: " << stack.peek() << std::endl;
70
71     return 0;
72 }
73
```

Output –

```
Output Clear  
/tmp/gLwCBjnFet.o  
Enter the capacity of the stack: 10  
Top element: 30  
Top element after pop: 20
```

9. Implement a program that uses dynamic memory allocation to simulate a banking system that stores customer information, account details, and transactions.

Code –



The screenshot shows a code editor window titled "main.cpp". The code is written in C++ and defines two structures: "Transaction" and "Account". The "Transaction" structure contains a string "type" and a double "amount". The "Account" structure contains an integer "accountNumber", a string "customerName", a double "balance", and a vector of "Transaction" objects named "transactions". There are two comments: one starting at line 17 and another at line 26. Lines 17 through 25 implement the "createAccount" function, which creates a new Account object and initializes its properties. Line 26 is a comment indicating the purpose of the function. The code editor has a dark theme with syntax highlighting for keywords and comments. It includes standard icons for file operations, brightness, and a "Run" button.

```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4
5 struct Transaction {
6     std::string type;
7     double amount;
8 };
9
10 struct Account {
11     int accountNumber;
12     std::string customerName;
13     double balance;
14     std::vector<Transaction> transactions;
15 };
16
17 // Function to create a new account
18 Account* createAccount(int accountNumber, const std::string& customerName) {
19     Account* newAccount = new Account;
20     newAccount->accountNumber = accountNumber;
21     newAccount->customerName = customerName;
22     newAccount->balance = 0.0;
23     return newAccount;
24 }
25
26 // Function to deposit money into an account
```

main.cpp

```
26 // Function to deposit money into an account
27 void deposit(Account* account, double amount) {
28     account->balance += amount;
29     Transaction transaction;
30     transaction.type = "Deposit";
31     transaction.amount = amount;
32     account->transactions.push_back(transaction);
33     std::cout << "Amount deposited successfully.\n";
34 }
35
36 // Function to withdraw money from an account
37 bool withdraw(Account* account, double amount) {
38     if (account->balance >= amount) {
39         account->balance -= amount;
40         Transaction transaction;
41         transaction.type = "Withdrawal";
42         transaction.amount = amount;
43         account->transactions.push_back(transaction);
44         std::cout << "Amount withdrawn successfully.\n";
45         return true;
46     } else {
47         std::cout << "Insufficient balance.\n";
48         return false;
49     }
50 }
51
```

main.cpp

```
52 // Function to display account information
53 void displayAccountInfo(const Account* account) {
54     std::cout << "Account Number: " << account->accountNumber << "\n";
55     std::cout << "Customer Name: " << account->customerName << "\n";
56     std::cout << "Account Balance: " << account->balance << "\n";
57 }
58
59 // Function to display transaction history
60 void displayTransactionHistory(const Account* account) {
61     std::cout << "Transaction History for Account Number " << account
62         ->accountNumber << ":\n";
63     for (const auto& transaction : account->transactions) {
64         std::cout << "Type: " << transaction.type << "\tAmount: " <<
65             transaction.amount << "\n";
66     }
67 }
68 // Function to delete an account and free its memory
69 void deleteAccount(Account* account) {
70     delete account;
71 }
72 int main() {
73     std::vector<Account*> accounts;
74
75     int accountNumber = 1001;
```

```
main.cpp

75     int accountNumber = 1001;
76     int choice;
77     do {
78         std::cout << "\nBanking System Menu:\n";
79         std::cout << "1. Create Account\n";
80         std::cout << "2. Deposit\n";
81         std::cout << "3. Withdraw\n";
82         std::cout << "4. View Account Information\n";
83         std::cout << "5. View Transaction History\n";
84         std::cout << "6. Exit\n";
85         std::cout << "Enter your choice: ";
86         std::cin >> choice;
87
88         switch (choice) {
89             case 1: {
90                 std::string customerName;
91                 std::cout << "Enter customer name: ";
92                 std::cin.ignore(); // Ignore the newline character left by
93                             // previous input
94                 std::getline(std::cin, customerName);
95
96                 Account* newAccount = createAccount(accountNumber,
97                                                 customerName);
98                 accounts.push_back(newAccount);
99                 std::cout << "Account created successfully. Account number: "
100                                << accountNumber << "\n";
101             }
102         }
103     }
104 }
```

main.cpp

```
98         << accountNumber << "\n";
99     accountNumber++;
100    break;
101 }
102 case 2: {
103     int accountNum;
104     double amount;
105     std::cout << "Enter account number: ";
106     std::cin >> accountNum;
107
108     Account* account = nullptr;
109     for (const auto& acc : accounts) {
110         if (acc->accountNumber == accountNum) {
111             account = acc;
112             break;
113         }
114     }
115     if (account == nullptr) {
116         std::cout << "Account not found.\n";
117         break;
118     }
119
120     std::cout << "Enter amount to deposit: ";
121     std::cin >> amount;
122     deposit(account, amount);
123 }
```

main.cpp

```
122         deposit(account, amount);
123         break;
124     }
125     case 3: {
126         int accountNum;
127         double amount;
128         std::cout << "Enter account number: ";
129         std::cin >> accountNum;
130
131         Account* account = nullptr;
132         for (const auto& acc : accounts) {
133             if (acc->accountNumber == accountNum) {
134                 account = acc;
135                 break;
136             }
137         }
138
139         if (account == nullptr) {
140             std::cout << "Account not found.\n";
141             break;
142         }
143
144         std::cout << "Enter amount to withdraw: ";
145         std::cin >> amount;
146         withdraw(account, amount);
147         break;
148     }
```

main.cpp

```
147         break;
148     }
149     case 4: {
150         int accountNum;
151         std::cout << "Enter account number: ";
152         std::cin >> accountNum;
153
154         Account* account = nullptr;
155         for (const auto& acc : accounts) {
156             if (acc->accountNumber == accountNum) {
157                 account = acc;
158                 break;
159             }
160         }
161
162         if (account == nullptr) {
163             std::cout << "Account not found.\n";
164             break;
165         }
166
167         displayAccountInfo(account);
168         break;
169     }
170     case 5: {
171         int accountNum;
172         std::cout << "Enter account number: ";
```

```
main.cpp | Run
```

```
172         std::cout << "Enter account number: ";
173         std::cin >> accountNum;
174
175         Account* account = nullptr;
176         for (const auto& acc : accounts) {
177             if (acc->accountNumber == accountNum) {
178                 account = acc;
179                 break;
180             }
181         }
182
183         if (account == nullptr) {
184             std::cout << "Account not found.\n";
185             break;
186         }
187
188         displayTransactionHistory(account);
189         break;
190     }
191     case 6:
192         // Free memory for all accounts
193         for (const auto& account : accounts) {
194             deleteAccount(account);
195         }
196         std::cout << "Exiting the program.\n";
197         break;
198 }
```

```
main.cpp | Run
```

```
181 }
182
183         if (account == nullptr) {
184             std::cout << "Account not found.\n";
185             break;
186         }
187
188         displayTransactionHistory(account);
189         break;
190     }
191     case 6:
192         // Free memory for all accounts
193         for (const auto& account : accounts) {
194             deleteAccount(account);
195         }
196         std::cout << "Exiting the program.\n";
197         break;
198     default:
199         std::cout << "Invalid choice. Try again.\n";
200     }
201 } while (choice != 6);
202
203 return 0;
204 }
```

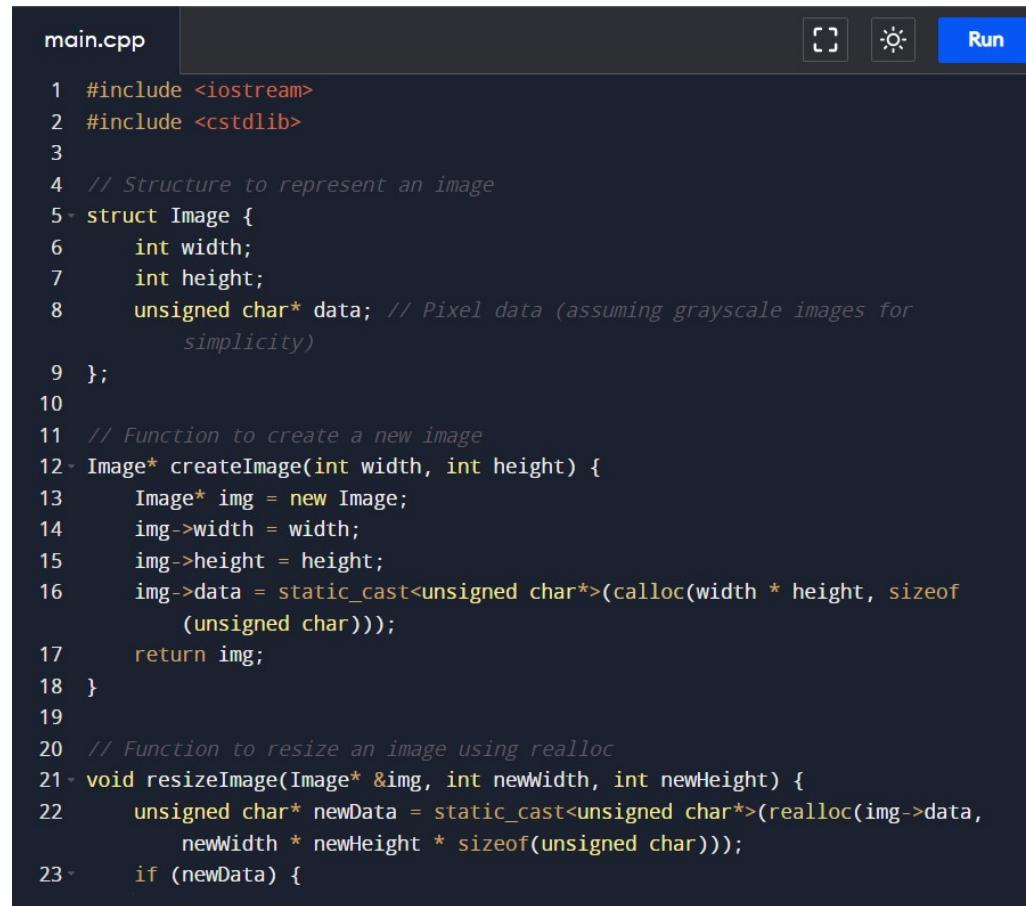
Output –

```
Output  
/tmp/VHYEiuzTAF.o  
Banking System Menu:  
1. Create Account  
2. Deposit  
3. Withdraw  
4. View Account Information  
5. View Transaction History  
6. Exit  
Enter your choice: 1  
Enter customer name: Adya  
Account created successfully. Account number: 1001  
  
Banking System Menu:  
1. Create Account  
2. Deposit  
3. Withdraw  
4. View Account Information  
5. View Transaction History  
6. Exit  
Enter your choice: 2  
Enter account number: 1001  
Enter amount to deposit: 5000  
Amount deposited successfully.  
  
Banking System Menu:  
1. Create Account
```

```
Output  
Amount deposited successfully.  
  
Banking System Menu:  
1. Create Account  
2. Deposit  
3. Withdraw  
4. View Account Information  
5. View Transaction History  
6. Exit  
Enter your choice: 3  
Enter account number: 1001  
Enter amount to withdraw: 2000  
Amount withdrawn successfully.  
  
Banking System Menu:  
1. Create Account  
2. Deposit  
3. Withdraw  
4. View Account Information  
5. View Transaction History  
6. Exit  
Enter your choice: 5  
Enter account number: 1001  
Transaction History for Account Number 1001:  
Type: Deposit Amount: 5000  
Type: Withdrawal Amount: 2000
```

10. Design a program that dynamically allocates memory for an image processing application, allowing users to resize and manipulate images.

Code –



The screenshot shows a code editor window with the file name "main.cpp" at the top left. At the top right, there are three icons: a square with two arrows, a sun-like icon, and a blue "Run" button. The code itself is written in C++ and defines a structure for an image and two functions: `createImage` and `resizeImage`.

```
main.cpp

1 #include <iostream>
2 #include <cstdlib>
3
4 // Structure to represent an image
5 struct Image {
6     int width;
7     int height;
8     unsigned char* data; // Pixel data (assuming grayscale images for
                           simplicity)
9 };
10
11 // Function to create a new image
12 Image* createImage(int width, int height) {
13     Image* img = new Image;
14     img->width = width;
15     img->height = height;
16     img->data = static_cast<unsigned char*>(calloc(width * height, sizeof
                           (unsigned char)));
17     return img;
18 }
19
20 // Function to resize an image using realloc
21 void resizeImage(Image* &img, int newWidth, int newHeight) {
22     unsigned char* newData = static_cast<unsigned char*>(realloc(img->data,
                           newWidth * newHeight * sizeof(unsigned char)));
23     if (newData) {
```

```
main.cpp Run

23     if (newData) {
24         img->data = newData;
25         img->width = newWidth;
26         img->height = newHeight;
27         std::cout << "Image resized to " << newWidth << "x" << newHeight << "."
28             << std::endl;
29     } else {
30         std::cout << "Memory reallocation failed. Image not resized." << std::endl;
31     }
32
33 // Function to release memory occupied by an image
34 void releaseImage(Image* &img) {
35     if (img) {
36         free(img->data);
37         delete img;
38         img = nullptr;
39         std::cout << "Image memory released." << std::endl;
40     }
41 }
42
43 int main() {
44     int initialWidth, initialHeight, newWidth, newHeight;
45
46     // Get initial image dimensions from the user
```

```
main.cpp
```

```
44     int initialWidth, initialHeight, newWidth, newHeight;
45
46     // Get initial image dimensions from the user
47     std::cout << "Enter initial image dimensions (width height): ";
48     std::cin >> initialWidth >> initialHeight;
49
50     // Create a new image
51     Image* image = createImage(initialWidth, initialHeight);
52     std::cout << "Image created with dimensions " << initialWidth << "x" <<
53                     initialHeight << "." << std::endl;
54
55     // Get new image dimensions for resizing from the user
56     std::cout << "Enter new image dimensions for resizing (width height): ";
57     std::cin >> newWidth >> newHeight;
58
59     // Resize the image
60     resizeImage(image, newWidth, newHeight);
61
62     // Perform more image processing operations...
63
64     // Release memory occupied by the image
65     releaseImage(image);
66
67     return 0;
68 }
```

Output –

```
Output Clear  
/tmp/Fzu6zWBjDY.o  
Enter initial image dimensions (width height): 640 480  
Image created with dimensions 640x480.  
Enter new image dimensions for resizing (width height): 800 600  
Image resized to 800x600.  
Image memory released.
```