Q1:

#include <iostream>

using namespace std;

int main() {

int n = 15;

int sum = 0;

for (int i = 1; i <= n; ++i) {

sum += i;

}

cout << "Sum of the first " << n << " natural numbers: " << sum << endl;

return 0;

}

Q1(B):

#include <iostream>

using namespace std;

int main()

{

int rows;

cout << "Enter the number of rows: ";

cin >> rows;

for (int i = 1; i <= rows; i++)

{

for (int j = 1; j <= rows-i; j++)

{

cout << " ";

}

for (int k = 1; k <= 2 \* i - 1 ; k++)

{

cout << "\*";

}

cout << endl;

}

return 0;

}

Q1©:

#include <iostream>

using namespace std;

int main()

{

int age;

cout << "Enter your age: ";

cin >> age;

if ( age >= 18 )

{

cout << "You'er eligible to vote.";

}

else if ( age <= 18 )

{

cout << "You'er not eligible to vote.";

}

else

{

cout << "You entered invalid input.";

}

}

Q1(D):

#include <iostream>

using namespace std;

int main()

{

int number, reversedNumber = 0, remainder;

cout << "Enter a number: ";

cin >> number;

do

{

remainder = number % 10;

reversedNumber = reversedNumber \* 10 + remainder;

number = number / 10;

} while (number != 0);

cout << "Reversed number: " << reversedNumber << endl;

return 0;

}

Q1(E);

#include <iostream>

#include <cctype>

using namespace std;

bool isPalindrome(const string& str)

{

string cleanedStr;

for (char ch : str)

{

if (isalnum(ch))

{

cleanedStr += tolower(ch);

}

}

int start = 0;

int end = cleanedStr.length() - 1;

while (start < end)

{

if (cleanedStr[start] != cleanedStr[end])

{

return false;

}

++start;

--end;

}

return true;

}

int main()

{

cout << "Enter a string: ";

string inputString;

getline(cin, inputString);

if (isPalindrome(inputString))

{

cout << "The entered string is a palindrome." << endl;

}

else

{

cout << "The entered string is not a palindrome." << endl;

}

return 0;

}

Q2(A):

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

int calculateDiagonalSum(const int matrix[][MAX\_SIZE], int size)

{

int sum = 0;

for (int i = 0; i < size; ++i)

{

sum += matrix[i][i];

}

return sum;

}

//a25762488@gmail.com

int main() {

int matrix[MAX\_SIZE][MAX\_SIZE];

int size;

cout << "Enter the size of the square matrix: ";

cin >> size;

cout << "Enter the elements of the matrix:" << endl;

for (int i = 0; i < size; ++i)

{

for (int j = 0; j < size; ++j)

{

cout << "Element at position (" << i + 1 << ", " << j + 1 << "): ";

cin >> matrix[i][j];

}

}

int diagonalSum = calculateDiagonalSum(matrix, size);

cout << "Sum of elements in the main diagonal: " << diagonalSum << endl;

return 0;

}

Q2(B):

#include <iostream>

using namespace std;

void selectionSort(int arr[], int size)

{

for (int i = 0; i < size - 1; ++i)

{

int minIndex = i;

for (int j = i + 1; j < size; ++j)

{

if (arr[j] < arr[minIndex])

{

minIndex = j;

}

}

swap(arr[i], arr[minIndex]);

}

}

int main()

{

int a[] = {125, 0, 695, 3, -256, -5, 214, 44, 55};

int size = sizeof(a) / sizeof(a[0]);

cout << "Original Array: ";

for (int i = 0; i < size; ++i)

{

cout << a[i] << " ";

}

selectionSort(a, size);

cout << "\nSorted Array: ";

for (int i = 0; i < size; ++i)

{

cout << a[i] << " ";

}

cout << endl;

return 0;

}

Q3(A):

#include <iostream>

using namespace std;

int fibonacci(int n)

{

if (n <= 1)

{

return n;

}

return fibonacci(n - 1) + fibonacci(n - 2);

}

int main()

{

cout << "Fibonacci Series (first 10 terms):" << endl;

for (int i = 0; i < 10; ++i)

{

cout << fibonacci(i) << " ";

}

cout << endl;

return 0;

}

Q3(B):

#include <iostream>

using namespace std;

double calculateAverage(const int arr[], int size)

{

double sum = 0.0;

for (int i = 0; i < size; ++i)

{

sum += arr[i];

}

return (size > 0) ? (sum / size) : 0.0;

}

int main()

{

const int maxSize = 100;

int arr[maxSize];

int size;

cout << "Enter the size of the array: ";

cin >> size;

cout << "Enter the elements of the array:" << endl;

for (int i = 0; i < size; ++i)

{

cout << "Element " << i + 1 << ": ";

cin >> arr[i];

}

double average = calculateAverage(arr, size);

cout << "Average of the elements: " << average << endl;

return 0;

}