

**National University of Computer and Emerging**

**Sciences**

**Chiniot-Faisalabad Campus BS (Artificial Intelligence)**

|  |  |
| --- | --- |
| **Name** | **M.Abdul Hanan** |
| **Reg.NO.** | **22F-3104** |
| **Section** | **BS(Ai)-3A1** |
| **Course** | **Data Structures** |
| **Department** | **CS Department** |
| **LAB** | **LAB Manual #7** |

**Task no 1:**

#include <iostream>

#include<string>

using namespace std;

// A recursive function that check palindrome or not.

bool isPal(char str[],int str1, int element)

{

// If there is only one character

if (str1 == element)

return true;

// If first and last

// characters do not match

if (str[str1] != str[element])

return false;

if (str1 < element + 1)

return isPal(str, str1 + 1, element - 1);

return true;

}

bool isPalindrome(char str[])

{

int n = strlen(str);// An empty string is considered as palindrome

if (n == 0)

return true;

return isPal(str, 0, n - 1);

}

int main()

{

while (true) {

char str[100];

cout << "Enter String :";

cin >> str;

if (isPalindrome(str))

cout << "Palindrome :"<<endl;

else

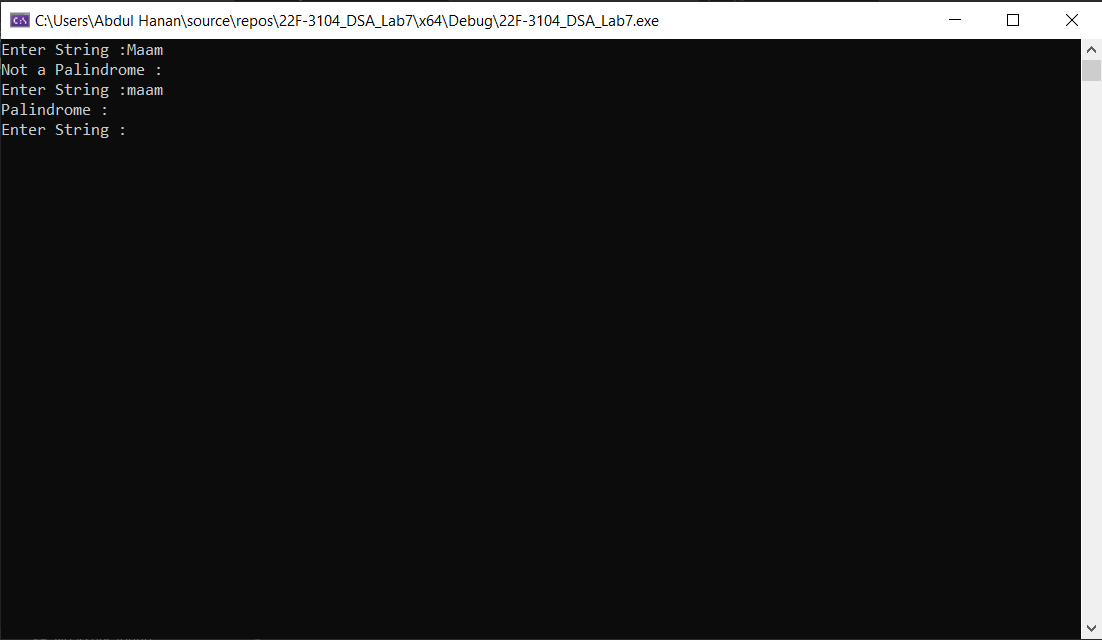
cout << "Not a Palindrome :"<<endl;

}

return 0;

}

**Screen Shot:**

****

**Task no 2:**

#include <iostream>

using namespace std;

int GCD(int num1, int num2)// Function to calculate the Greatest Common Divisor

{

if (num2 != 0)

return GCD(num2, num1 % num2);// Recursive call with swapped arguments and remainder

else

return num1;// If n2 becomes 0 return n1 as the GCD

}

int main()

{

int num1, num2;

while (true) {

cout << "Enter integer 1: ";

cin >> num1;

cout << "Enter integer 2: ";

cin >> num2;

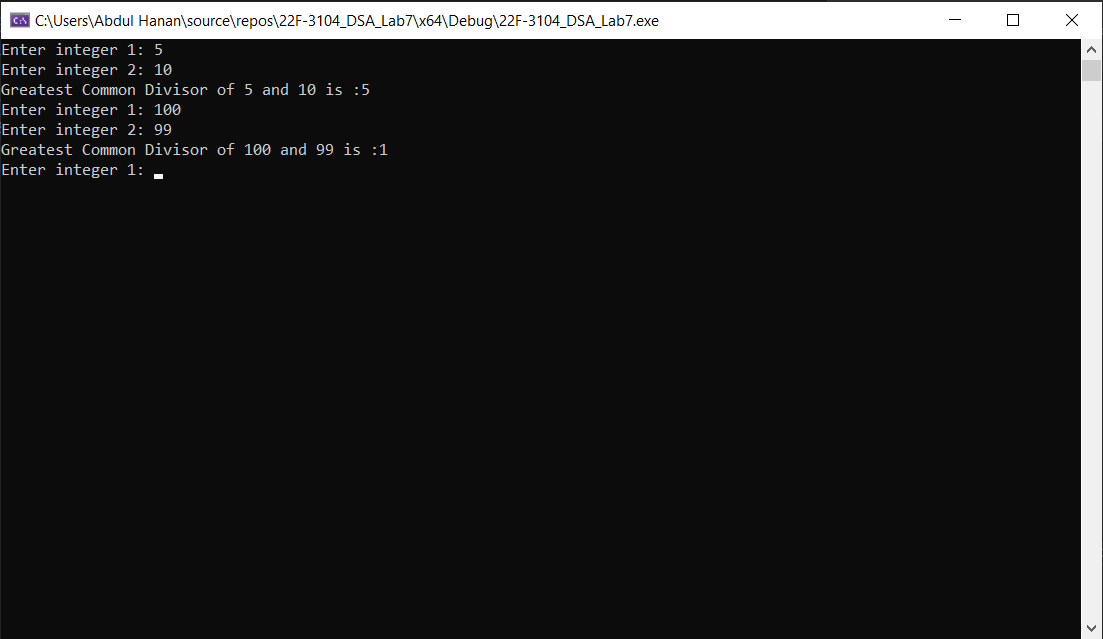
cout << "Greatest Common Divisor of " << num1 << " and " << num2 << " is :" << GCD(num1, num2)<<endl;

}

return 0;

}

**Screen Shot:**

****

**Task no 3:**

#include <iostream>

using namespace std;

// Function to check if a number is binary

bool isBinary(long n) {

while (n > 0) {

int digit = n % 10;

if (digit != 0 && digit != 1) {

return false; // It is not a binary number

}

n /= 10;

}

return true; // It is a binary number

}

// Function to convert a binary number to decimal

int BinaryToDecimal(long n) {

int dec = 0, i = 0, rem;

while (n != 0) {

rem = n % 10;

n /= 10;

dec += rem \* pow(2, i);

++i;

}

return dec;

}

int main() {

long n;

while (true) {

cout << "Enter a binary number: ";

cin >> n;

if (isBinary(n)) {

cout << "The Decimal of " << n << " Binary is : " << BinaryToDecimal(n) << endl;

}

else {

cout << "Not a binary number." << endl;

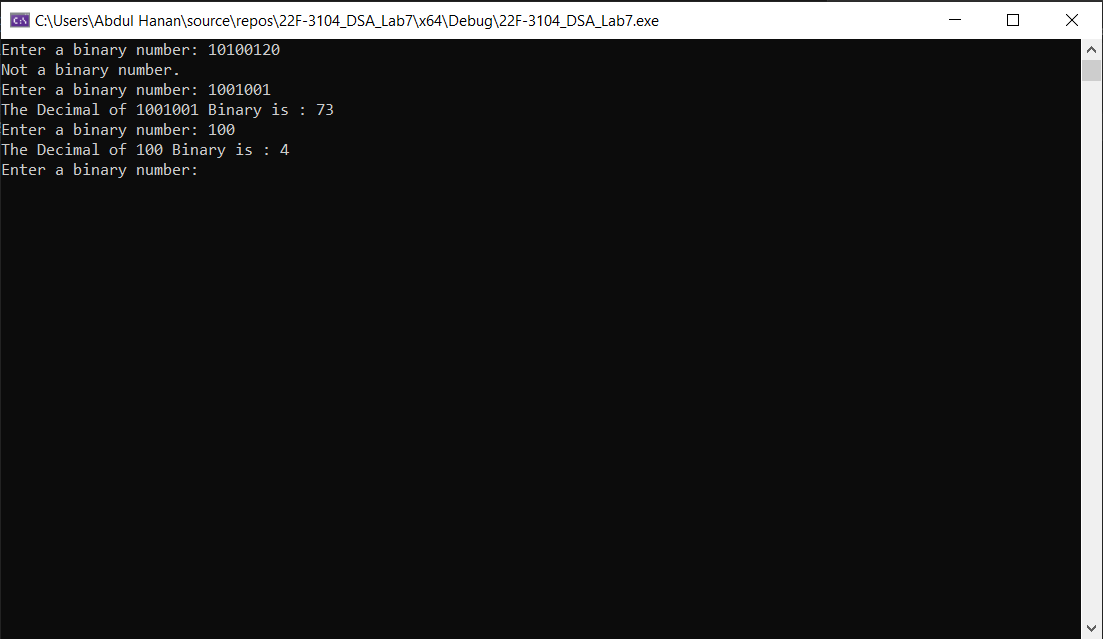
}

}

return 0;

}

**Screen Shot:**

****

**Task no 4:**

#include <iostream>

#include <string>

using namespace std;

// Function to find and return the first uppercase character in a string

char first(string str) {

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

// check if the current character is a letter and is uppercase

if (str[i] >= 'A' && str[i] <= 'Z') {

return str[i]; // return the first uppercase letter found

}

}

return 0; // return 0 if no uppercase letter was found

}

int main() {

string str;

cout << "Enter a string: ";

cin >> str;

char res = first(str);

if (res == 0) {

cout << "No uppercase letter" << endl;

}

else {

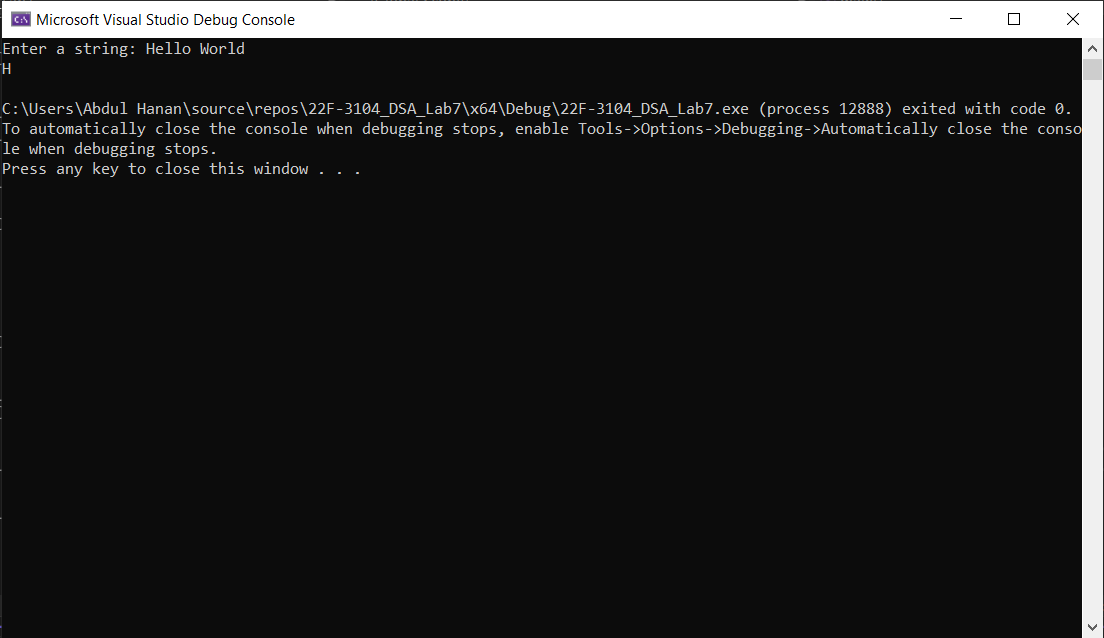
cout << res << endl;

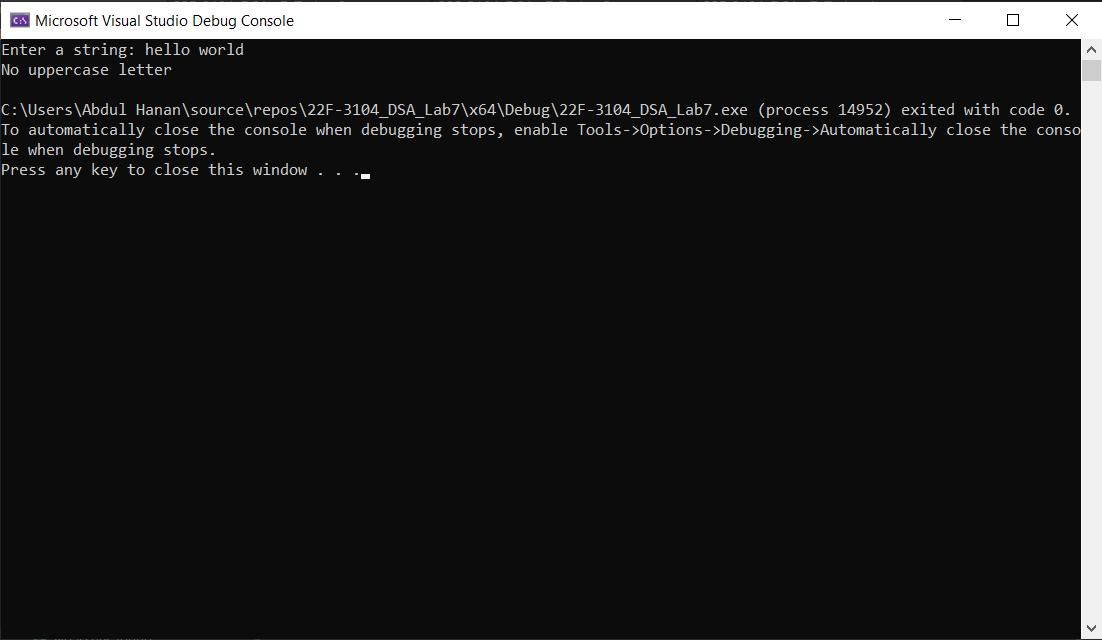
}

return 0;

}

**Screen Shot:**





**Task no 5:**

#include <iostream>

using namespace std;

// Function to print Minimum element using recursion

int findMini(int\* arr, int size)

{

// if size = 1 means the whole array has to be traverse

if (size == 1)

return arr[0];

// Find the minimum of the remaining elements

int miniRemain = findMini(arr, size - 1);

return (arr[size - 1] < miniRemain) ? arr[size - 1] : miniRemain;

}

int main()

{

int size;

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

cin >> size;

int\* arr = new int[size];

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

cout << "Enter index number of array " << i + 1 << ": ";

cin >> arr[i];

}

int minNum = findMini(arr, size);

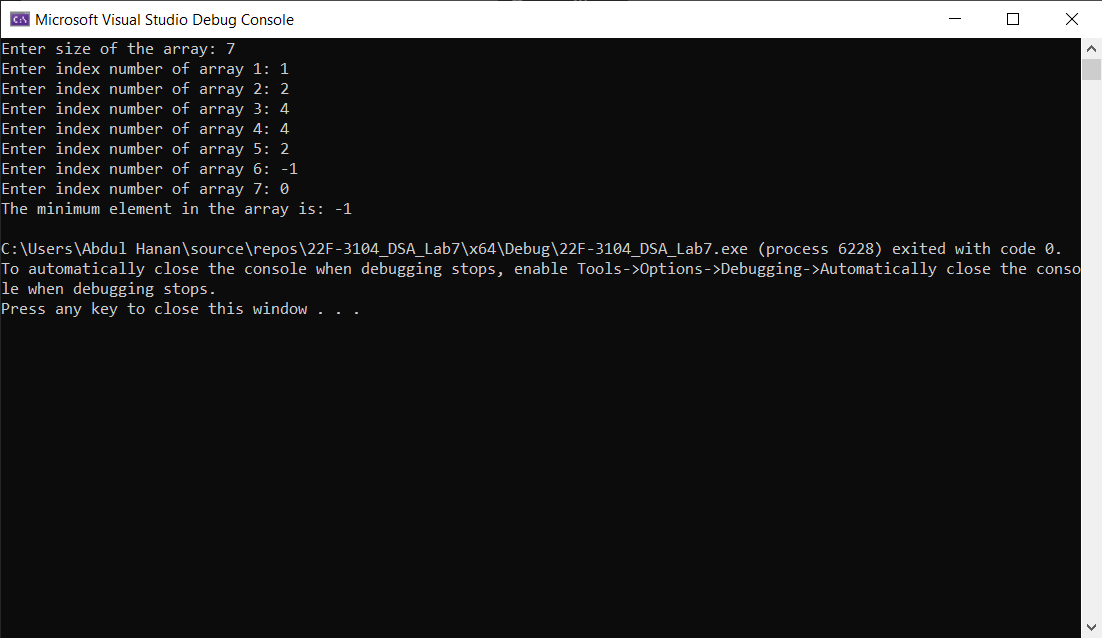
cout << "The minimum element in the array is: " << minNum << endl;

delete[] arr;

return 0;

}

**Screen shot:**

****

**Task no 6:**

#include <iostream>

#include <string>

using namespace std;

// Function to check if a word is elfish

bool elfish(const string word, int index, bool hasE, bool hasL, bool hasF) {

// If we have checked the entire word return true if it is elfish

if (index == word.length()) {

return hasE && hasL && hasF;

}

// Check the current character

char currentChar = word[index];

if (currentChar == 'e') {

hasE = true;

}

else if (currentChar == 'l') {

hasL = true;

}

else if (currentChar == 'f') {

hasF = true;

}

// Recursively check the rest of the word

return elfish(word, index + 1, hasE, hasL, hasF);

}

int main() {

string word;

while (true) {

cout << "Enter a word: ";

cin >> word;

bool isElfish = elfish(word, 0, false, false, false);

if (isElfish) {

cout << word << " is elfish." << endl;

}

else {

cout << word << " is not elfish." << endl;

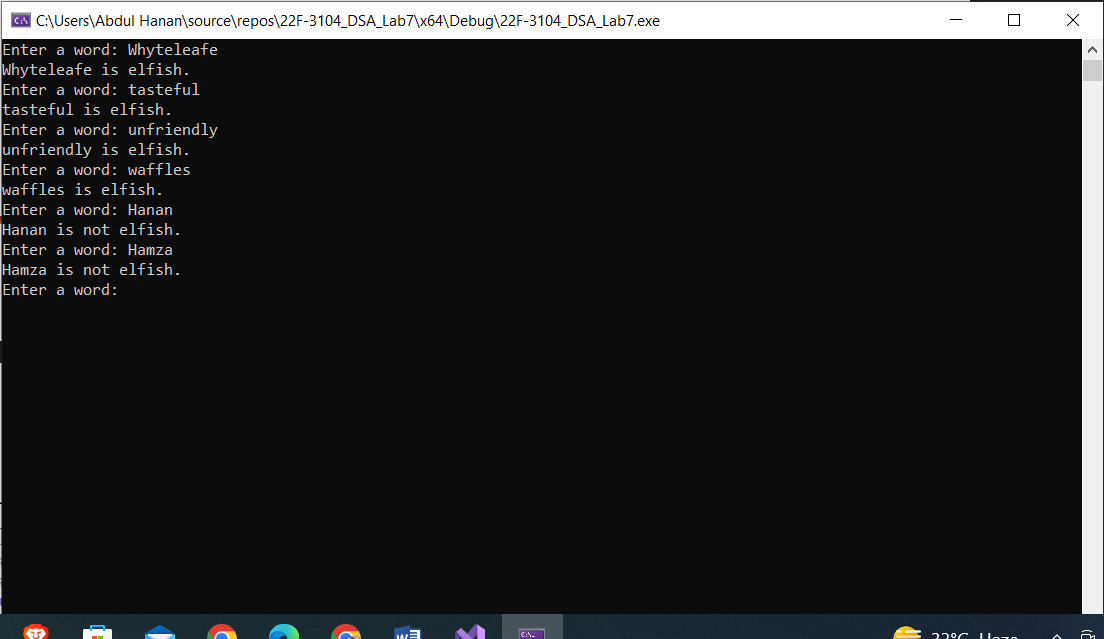
}

}

return 0;

}

**Screen shot:**

****