LAB MANUAL

PROGRAMMING WITH C++

WEEK 1 TO WEEK 14

SUBMITTED TO: Dr. Manzoor Ahmad

2024

MOHIZEEN AHMAD PARA

MCA Semi-1st

2404511052

GUIDING OUR PATH TO A BRIGHT FUTURE

LAB EXERCISES

Week 1

Q1. Write a program to demonstrate the use of output statements that draws any object of your choice **a. Draw a Christmas tree using '*'**
Q2. Write a program that reads in a month number and outputs the month name.
Q3. Write a program to demonstrate the use of various input statements like getchar(), getch(),scanf()
Q4. Write a program to demonstrate the overflow and underflow of various data type and their resolution

In this week

>> Overflow occurs when a calculation produces a result that is greater than the maximum value the data type can hold.

>> Underflow occurs when a calculation produces a result that is smaller than the minimum value the data type can hold.

Q1. Write a program to demonstrate the use of output statements that draws any object of your choice

- **Christmas Tree**

SOURCE CODE

```
#include<stdio.h>                                   // Include standard input-output library
int main()                                          // Main function where execution begins
{
   int i, l, k, m, j, rows, starNo, spaceNo;        // Declare integer variables
   printf("Enter No. of Rows Length of tree:\n"); // Prompt user to enter the number of rows
   scanf("%d", &rows);                              // Read the number of rows from user input
   for(i = 1; i <= rows; i++)                       // Loop through each row
   {S
      starNo = i * 2 - 1;                           // Calculate the number of stars for the current row
      spaceNo = i + rows - starNo;                  // Calculate the number of spaces before stars
      for(j = 0; j < spaceNo; j++)                  // Loop to print spaces
      {
         printf(" ");                               // Print a space
      }
      for(k = 0; k < starNo; k++)                   // Loop to print stars
      {
         printf("*");                               // Print a star
      }
      printf("\n");                                 // Move to the next line after printing stars
   }
   for(l = 0; l < 3; l++)                           // Loop to print the trunk of the tree
   {
      for(m = 0; m < (rows * 2 + 1) / 2; m++)       // Loop to print spaces before the trunk
      {
         printf(" ");                               // Print a space
      }
      printf("*\n");                                // Print the trunk of the tree
   }
   return 0;                                        // Return 0 to indicate successful execution
}
```

OUTPUT

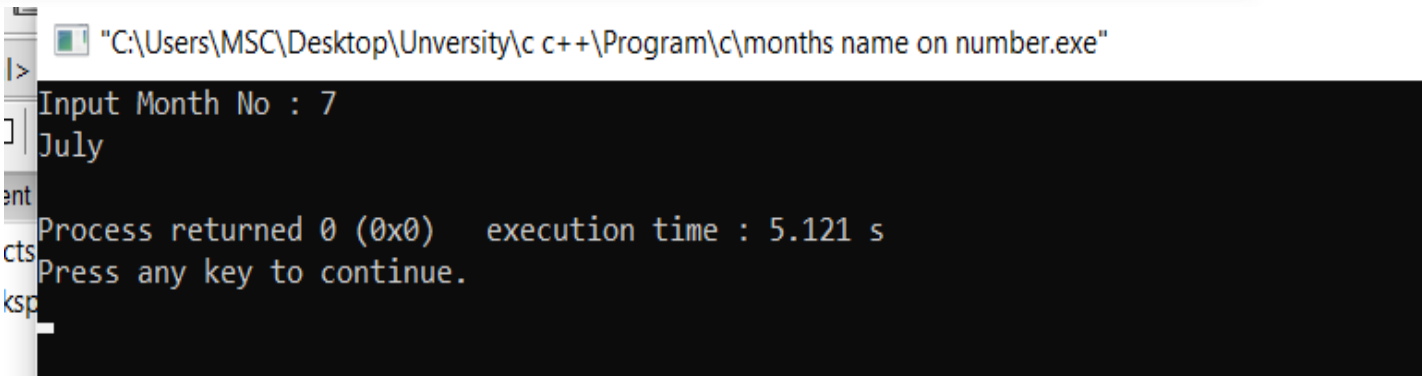
```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\chrismai tree.exe"
Enter No. of Rows Length of tree:
7
        *
       ***
      *****
     *******
    *********
   ***********
  *************
        *
        *
        *

Process returned 0 (0x0)   execution time : 4.100 s
Press any key to continue.
```

Q2. Write a program that reads in a month number and outputs the month name.

SOURCE CODE

```
#include <stdio.h>
int main()
{
   int monno;                                  // Declare an integer variable to store the month number
   printf("Input Month No : ");                // Prompt user to input the month number
   scanf("%d", &monno);                        // Read the month number from user input
   switch(monno)                               // Start of switch-case to check the month number
   {
      case 1:
         printf("January\n");                 // Print "January" if month number is 1
         break;                               // Break out of the switch-case
      case 2:
         printf("February\n");                // Print "February" if month number is 2
         break;                               // Break out of the switch-case
      case 3:
         printf("March\n");                   // Print "March" if month number is 3
         break;                               // Break out of the switch-case
      case 4:
         printf("April\n");                   // Print "April" if month number is 4
         break;                               // Break out of the switch-case
      case 5:
         printf("May\n");                     // Print "May" if month number is 5
         break;                               // Break out of the switch-case
      case 6:
         printf("June\n");                    // Print "June" if month number is 6
         break;                               // Break out of the switch-case
      case 7:
         printf("July\n");                    // Print "July" if month number is 7
         break;                               // Break out of the switch-case
      case 8:
         printf("August\n");                  // Print "August" if month number is 8
         break;                               // Break out of the switch-case
      case 9:
         printf("September\n");               // Print "September" if month number is 9
         break;                               // Break out of the switch-case
      case 10:
         printf("October\n");                 // Print "October" if month number is 10
         break;                               // Break out of the switch-case
      case 11:
         printf("November\n");                // Print "November" if month number is 11
         break;                               // Break out of the switch-case
      case 12:
         printf("December\n");                // Print "December" if month number is 12
         break;                               // Break out of the switch-case
      default:
         printf("Invalid Month number\n");    // Print error message if month number is not between 1 and 12
         break;                               // Break out of the switch-case
   }
   return 0;                                   // Return 0 to indicate successful execution
}
```

OUTPUT

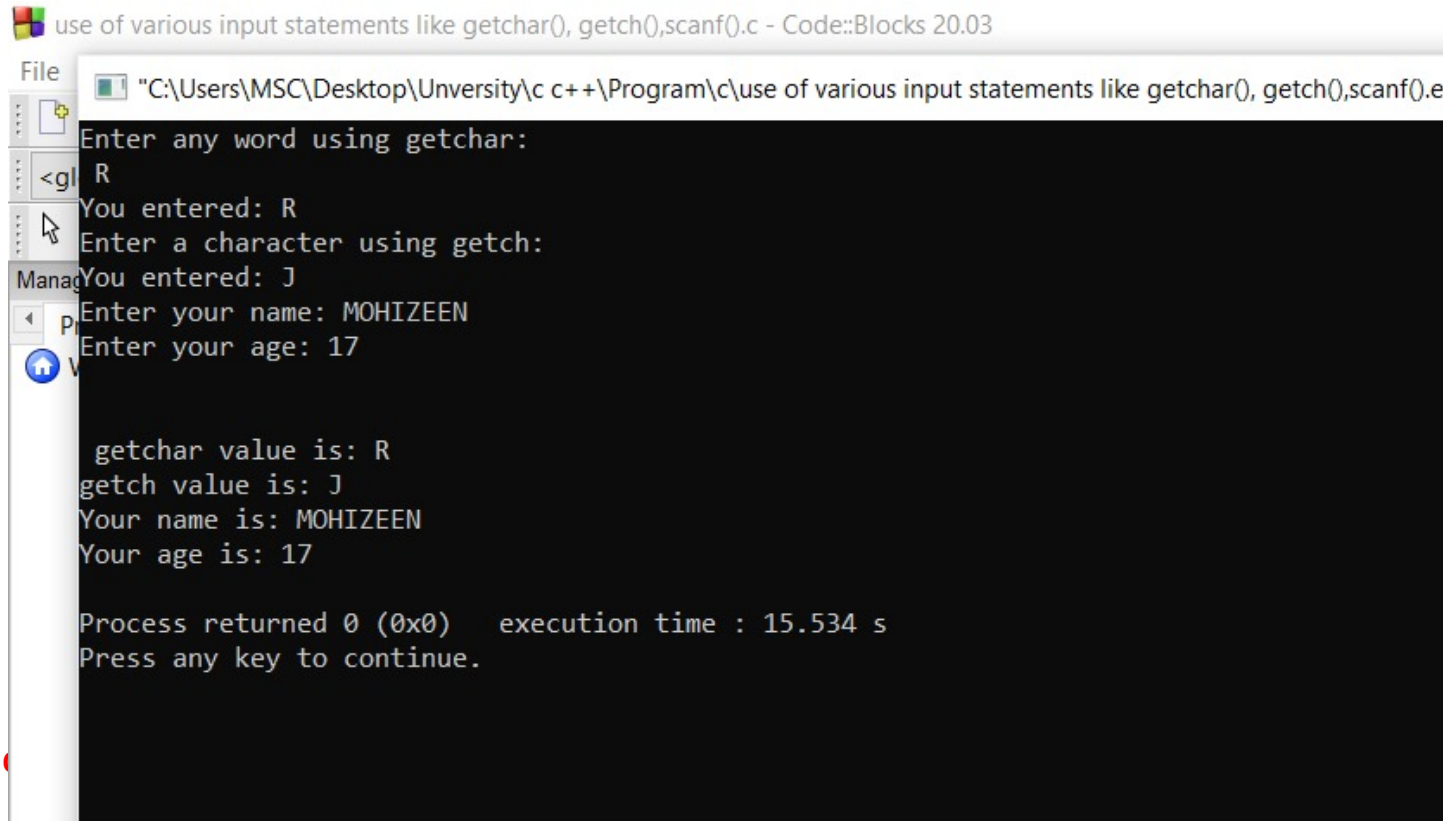


Q3. Write a program to demonstrate the use of various input statements like getchar(), getch(),scanf()

SOURCE CODE

```
#include <stdio.h>
#include <conio.h>                          // Include conio.h for getch()

int main() {
   char ch1, ch2;
   char name[50];
   int age;
   printf("Enter a word using getchar:\n ");      // Using getchar()
   ch1 = getchar();
   getchar();                                        // Consume the newline character left by getchar()
   printf("You entered: %c\n", ch1);
   printf("Enter a character using getch: ");          // Using getch()
   ch2 = getch();
   printf("\nYou entered: %c\n", ch2);
   printf("Enter your name: ");                      // Using scanf()
   scanf("%s", name);
   printf("Enter your age: ");
   scanf("%d", &age);
   printf("\n \n getchar value is: %c\n", ch1);              // Corrected printf statements
   printf("getch value is: %c\n", ch2);
   printf("Your name is: %s\n", name);
   printf("Your age is: %d\n", age);

   return 0;
}
```

OUTPUT

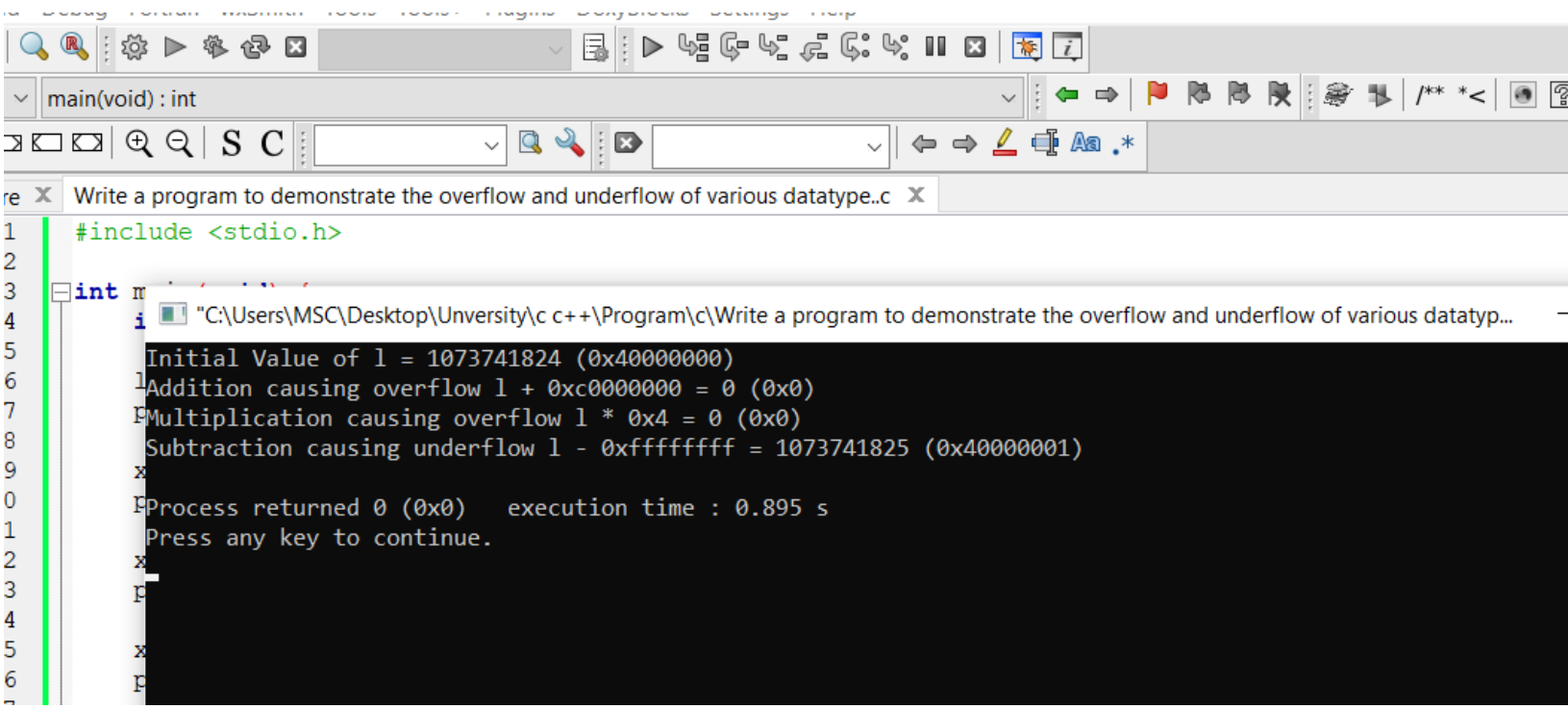


SOURCE CODE

Note:- overflow and underflow in integer datatype

```
#include <stdio.h>
int main(void) {
   int l, x;

   l = 0x40000000;                                    // Initial Value of l
   printf("Initial Value of l = %d (0x%x)\n", l, l);  // Print the initial value of l in decimal and hexadecimal

   x = l + 0xc0000000;                                // Addition causing overflow
   printf("Addition causing overflow l + 0xc0000000 = %d (0x%x)\n", x, x);   // Print the result in decimal and hexadecimal

   x = l * 0x4;                                       // Multiplication causing overflow
   printf("Multiplication causing overflow l * 0x4 = %d (0x%x)\n", x, x);   // Print the result in decimal and hexadecimal

   x = l - 0xffffffff;                                // Subtraction causing underflow
   printf("Subtraction causing underflow l - 0xffffffff = %d (0x%x)\n", x, x);   // Print the result in decimal and hexadecimal

   return 0;
}
```

OUTPUT



LAB EXERCISES

Week 2

Q5. Write a program to demonstrate the precedence of various operators
Q6. Write a program to generate a sequence of numbers in both ascending and descending order.
Q7. Write a program to generate pascals triangle
Q8. Write a program to reverse the digits of a given number.

In this week

>> PASCAL'S TRIANGLE

Pascal's Triangle is a triangular array of numbers where each number is the sum of the two numbers directly above it. It's named after the French mathematician Blaise Pascal, although it was known to mathematicians in India, China, and Persia long before his time.

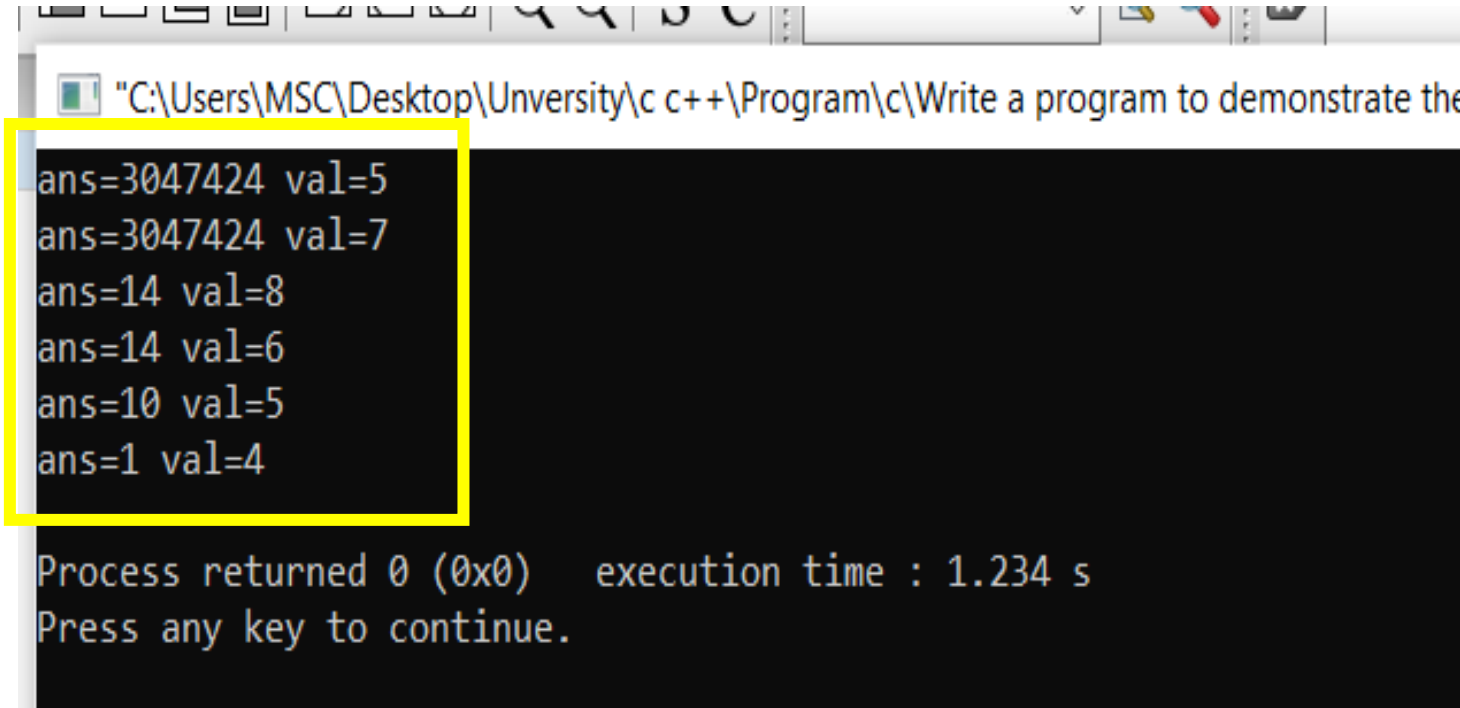
Q5. Write a program to demonstrate the precedence of various operators

SOURCE CODE

Note:- overflow and underflow in integer datatype

```
#include <stdio.h>
int main() {
   int ans, val = 4;
   val = val + 1;                          // Increment val by 1
   printf("ans=%d val=%d\n", ans, val);
   val++;                                  // Post-increment
   ++val;                                  // Pre-increment
   printf("ans=%d val=%d\n", ans, val);
   ans = 2 * val++;                        // Multiplication has higher precedence than post-increment
   printf("ans=%d val=%d\n", ans, val);
   val--;                                  // Post-decrement
   --val;                                  // Pre-decrement
   printf("ans=%d val=%d\n", ans, val);
   ans = --val * 2;                        // Pre-decrement has higher precedence than multiplication
   printf("ans=%d val=%d\n", ans, val);
   ans = val-- / 3;                        // Division has higher precedence than post-decrement
   printf("ans=%d val=%d\n", ans, val);
   return 0;
}
```

OUTPUT



Q6. Write a program to generate a sequence of numbers in both ascending and descending order.

SOURCE CODE

```
#include <stdio.h>
int main() {
  int n, data[100], i, j, temp;
    printf("Enter your Size of Intergers order: ");                    // Get the number of entries
  scanf("%d", &n);
  for (i = 1; i <= n; i++) {                                            // Get the input sequence
        printf("Enter %d No. value between 0-9:\n",i);
    scanf("%d", &data[i]);
  }
  for (i = 0; i < n - 1; i++) {                                         // Sort the sequence in ascending order
    for (j = i + 1; j < n; j++) {
      if (data[i] > data[j]) {
        temp = data[i];
        data[i] = data[j];
        data[j] = temp;
      } } }
    printf("Ascending Order:\n");                          // Print the sequence in ascending order
  for (i = 0; i < n; i++) {
    printf("%d\n", data[i]);
  }
   printf("\nDescending Order:\n");                         // Print the sequence in descending order
  for (i = n - 1; i >= 0; i--) {
    printf("%d\n", data[i]);
  }
  return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\Write a program t
Enter your Size of Intergers order: 7
Enter 1 No. value between 0-9:
2
Enter 2 No. value between 0-9:
4
Enter 3 No. value between 0-9:
7
Enter 4 No. value between 0-9:
5
Enter 5 No. value between 0-9:
1
Enter 6 No. value between 0-9:
0
Enter 7 No. value between 0-9:
6
Ascending Order:
0
0
1
2
4
5
7

Descending Order:
7
5
4
2
1
0
0

Process returned 0 (0x0)   execution time : 10.422 s
Press any key to continue.
```

Q7. Write a program to generate pascals triangle.

SOURCE CODE

```
#include <stdio.h>
long factorial(int n) {                    // Function to calculate factorial
   int c;
   long result = 1;
   for (c = 1; c <= n; c++) {
      result = result * c;                 // Calculate factorial
   }
   return result;                          // Return the factorial result
}
int main() {
   int i, n, c;
   printf("Enter the number of rows you wish to see in Pascal's Triangle\n");
   scanf("%d", &n);                                  // Read the number of rows from user input
   for (i = 0; i < n; i++) {
      for (c = 0; c <= (n - i - 2); c++) {
         printf(" ");                                // Print spaces for formatting
      }
      for (c = 0; c <= i; c++) {
          printf("%ld ", factorial(i) / (factorial(c) * factorial(i - c)));   // Calculate and print the binomial coefficient
      }
      printf("\n");                                  // Move to the next line after printing each row
   }
   return 0;  // Return 0 to indicate successful execution
}
```

OUTPUT

lobal> | factorial(int n) : long

"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\Write a program to generate pascals triangle.exe"

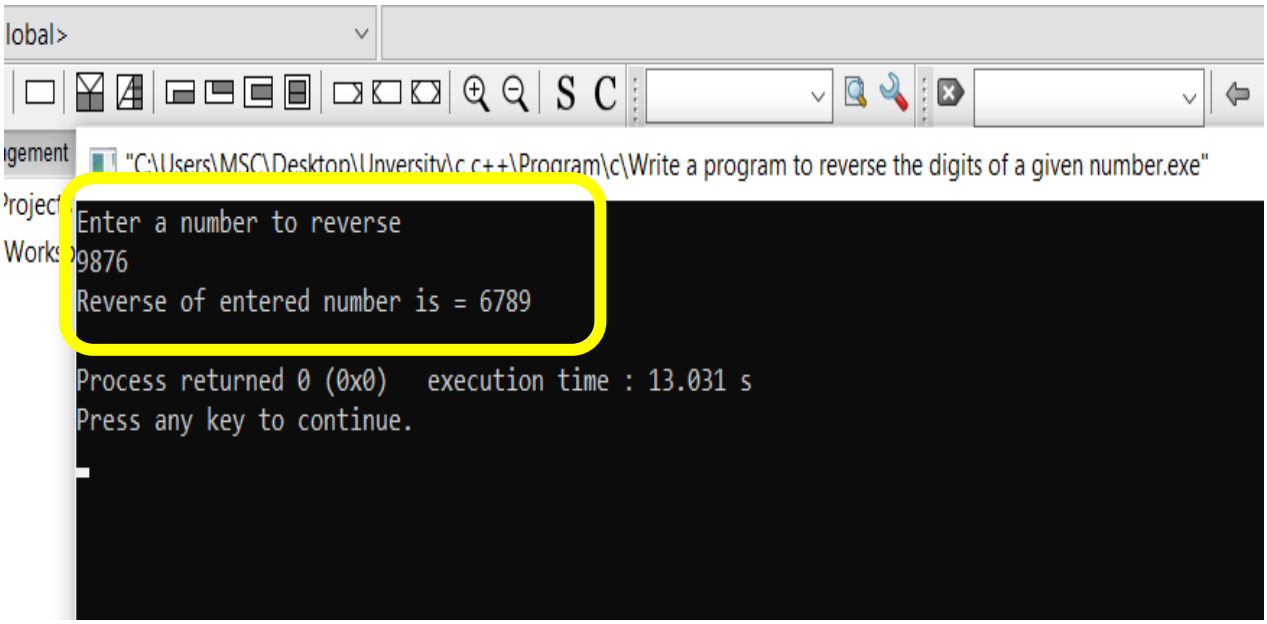
```
Enter the number of rows you wish to see in Pascal's Triangle
7
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1

Process returned 0 (0x0)   execution time : 5.063 s
Press any key to continue.
```

Q8. Write a program to reverse the digits of a given number.

SOURCE CODE

```
#include <stdio.h>  // Include standard input-output library


int main() {
   int n, reverse = 0;  // Declare integer variables


   printf("Enter a number to reverse\n");  // Prompt user to enter a number
   scanf("%d", &n);  // Read the number from user input


   while (n != 0) {  // Loop until n becomes 0
      reverse = reverse * 10;  // Multiply reverse by 10
      reverse = reverse + n % 10;  // Add the last digit of n to reverse
      n = n / 10;  // Remove the last digit from n
   }


   printf("Reverse of entered number is = %d\n", reverse);  // Print the reversed number
   return 0;  // Return 0 to indicate successful execution
}
```

OUTPUT



LAB EXERCISES

Week 3

Q9. Write a program to convert an amount in figures to equivalent amount in words
Q10. Write a program to find sum of all prime numbers between 100 and 500.
Q11. Create a one dimensional array of characters and store a string inside it by reading from standard input.
Q12. Write a program to input 20 arbitrary numbers in one dimensional array. Calculate the frequency of each number. Print the number and its frequency in a tabular form

In this week

1 - One
10 - Ten
100 - Hundred
1,000 - Thousand
10,000 - Ten Thousand
100,000 - Hundred Thousand
1,000,000 - Million
10,000,000 - Ten Million
100,000,000 - Hundred Million
1,000,000,000 - Billion
10,000,000,000 - Ten Billion
100,000,000,000 - Hundred Billion
1,000,000,000,000 - Trillion
10,000,000,000,000 - Ten Trillion
100,000,000,000,000 - Hundred Trillion

Q9. Write a program to convert an amount in figures to equivalent amount in words

SOURCE CODE

```
#include<stdio.h>                                   // Include standard input-output library
void pw(long, char[]);                              // Function to print words for a given number
 char *one[] = {"", "one", "two", "three", "four", "five", "six", "seven",
          "eight", "nine", "ten", "eleven", "twelve", "thirteen", "fourteen",   // Arrays to store words for numbers
          "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};
char *ten[] = {"", "", "twenty", "thirty", "forty", "fifty", "sixty",
          "seventy", "eighty", "ninety"};

int main() {
   long n;
   printf("Enter any 9 digit number: ");                    // Prompt user to enter a number
   scanf("%9ld", &n);                                       // Read the number from user input

   if (n <= 0) {
      printf("Enter numbers greater than 0\n");                    // Check if the number is greater than 0
   } else {
      pw((n / 10000000), "crore");                      // Convert and print the crore part of the number
      pw(((n / 100000) % 100), "lakh");                 // Convert and print the lakh part of the number
      pw(((n / 1000) % 100), "thousand");               // Convert and print the thousand part of the number
      pw(((n / 100) % 10), "hundred");                  // Convert and print the hundred part of the number
      pw((n % 100), "");                                // Convert and print the remaining part of the number
   }
   return 0;
}                                                       // Function to print words for a given number
void pw(long n, char ch[]) {
   (n > 19) ? printf("%s %s ", ten[n / 10], one[n % 10]) : printf("%s ", one[n]);        // Convert and print the number in words
   if (n) printf("%s ", ch);                        // Print the corresponding word (crore, lakh, etc.) if the number is not zero
}
```

OUTPUT

```
14      long n;
15      printf("Enter any 9 digit number: ");  // Prompt user to enter a number
```

"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\Write a program to convert an amount in figures to equivalent amount in words....

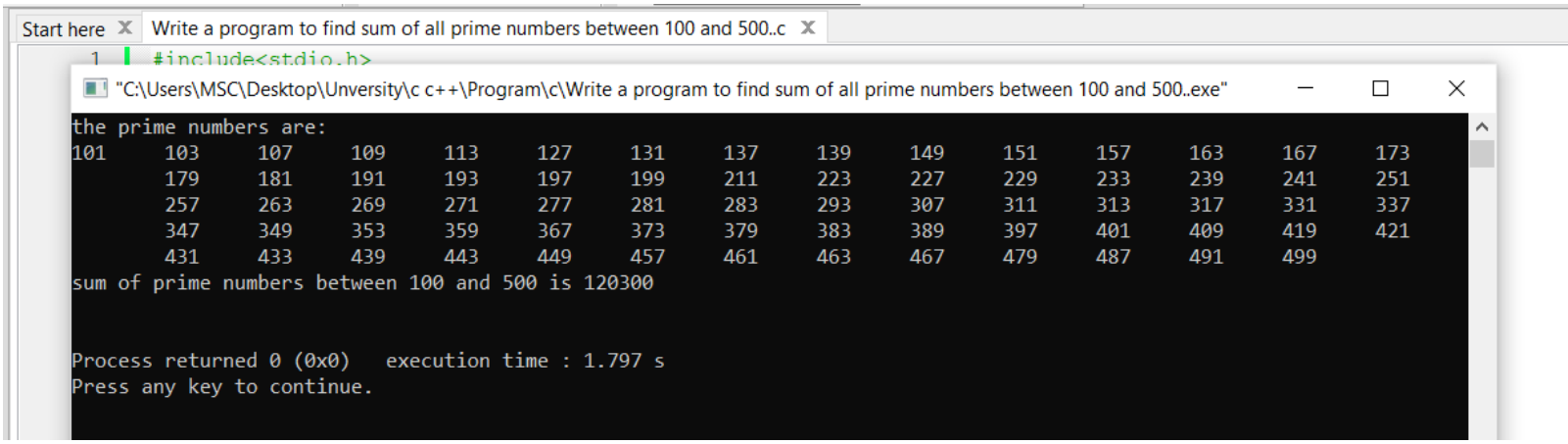
```
Enter any 9 digit number: 705147985
seventy  crore fifty one lakh forty seven thousand nine hundred eighty five
Process returned 0 (0x0)   execution time : 9.141 s
Press any key to continue.
```

Q10. Write a program to find sum of all prime numbers between 100 and 500.

SOURCE CODE

```
#include<stdio.h>                                   // Include standard input-output library

int main() {
   int i, j, sum = 0;                                // Declare integer variables

   printf("The prime numbers are:\n");              // Print a message

   for (i = 100; i <= 500; i++) {                   // Loop through numbers from 100 to 500
      for (j = 2; j < i; j++) {                 // Check if the number is divisible by any number less than itself
         if (i % j == 0)                        // If the number is divisible, it's not a prime number
            break;
      }
      if (i == j)                          // If the number is only divisible by itself, it's a prime number
         printf("%d\t", i);                // Print the prime number
      sum = sum + i;                       // Add the number to the sum
   }

   printf("\nSum of prime numbers between 100 and 500 is %d\n\n", sum);          // Print the sum of prime numbers
   return 0;                               // Return 0 to indicate successful execution
}
```

OUTPUT

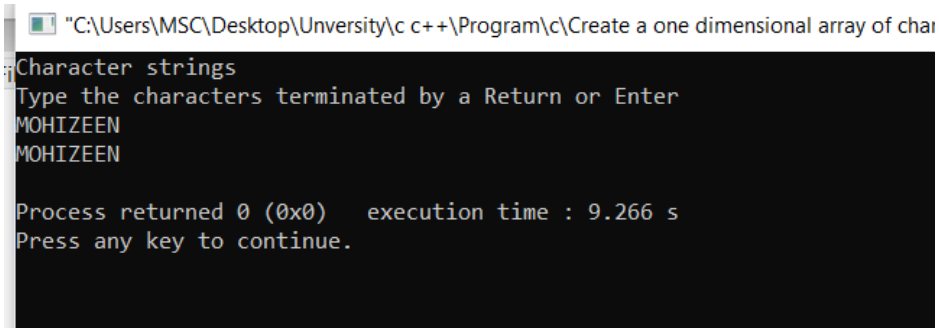


Q11. Create a one dimensional array of characters and store a string inside it by reading from standard input

SOURCE CODE

```
#include<stdio.h>
int main() {
   char msg[50], ch;                    // Declare a character array and a character variable
   int i = 0;                           // Initialize an integer variable
   printf("Character strings\n");       // Print a message
   printf("Type the characters terminated by a Return or Enter\n");          // Prompt user to type characters

   while ((ch = getchar()) != '\n')     // Read characters until a newline is encountered
      msg[i++] = ch;                    // Store each character in the array
   msg[i] = '\0';                       // Null-terminate the string

   i = 0;                               // Reset the index variable
   while (msg[i] != '\0')               // Loop through the array until the null character is encountered
      putchar(msg[i++]);                // Print each character

   printf("\n");                        // Print a newline character
   return 0;                            // Return 0 to indicate successful execution
}
```

OUTPUT

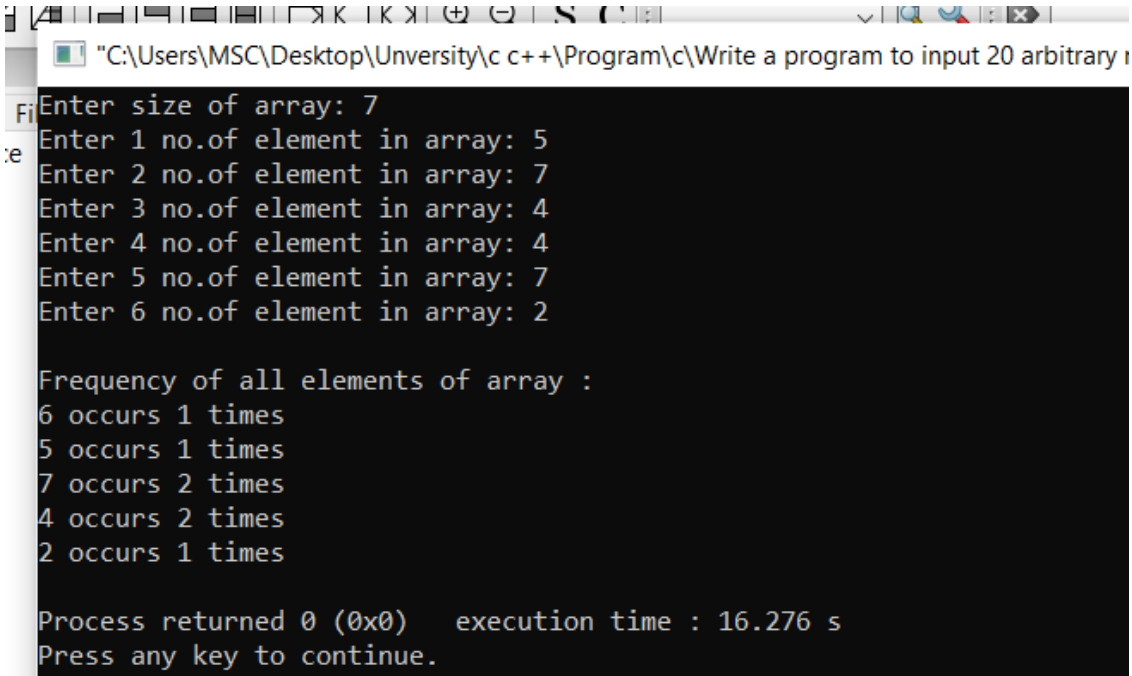


Q12. Write a program to find sum of all prime numbers between 100 and 500.

SOURCE CODE

```
#include <stdio.h>
int main()
{
int arr[100], freq[100];
int size, i, j, count;                    /* Input size of array */
printf("Enter size of array: ");
scanf("%d", &size);
for(i=1; i<+size; i++)                    /* Input elements in array */
{
 printf("Enter %d no.of element in array: ",i);
scanf("%d", &arr[i]);
freq[i] = -1;                                     /* Initially initialize frequencies to -1 */
}
for(i=0; i<size; i++)
{
count = 1;
for(j=i+1; j<size; j++)
{
if(arr[i]==arr[j])                                /* If duplicate element is found */
{
count++;
freq[j] = 0;                                      /* Make sure not to count frequency of same element again */
}
}
if(freq[i] != 0)                                  /* If frequency of current element is not counted */
{
freq[i] = count;
}
}
printf("\nFrequency of all elements of array : \n");                //Print frequency of each element
for(i=0; i<size; i++)
{
if(freq[i] != 0)
{
printf("%d occurs %d times\n", arr[i], freq[i]);
}
}
return 0;
}
```

OUTPUT



LAB EXERCISES

Week 4

Q13. Write a C function to remove duplicates from an ordered array.
Q14. Write a program which will arrange the positive and negative numbers in one dimensional array in such a way that all negative numbers should come first and then all the positive numbers will come without changing the original sequence of numbers.
Q15. Write a program to compute Addition, Multiplication, Transpose on 2D array.
Q16. Implement a program which uses multiple files for holding multiple functions which are compiled separately, linked together and called by main(). Use static and extern variables in these files .

In this week

Multi-File Program with Static and Extern Variables

This program demonstrates the use of multiple files to hold different functions, which are compiled separately and linked together. It uses both static and extern variables.

Files

1. **main.c**: Contains the main() function.
2. **file1.c**: Contains function definitions.
3. **file1.h**: Contains function declarations and extern variable declaration.

Q13. Write a program to convert an amount in figures to equivalent amount in words

SOURCE CODE

```
#include <stdio.h>

int main() {
   int n, a[100], b[100], count = 0, c, d;                    // Declare integer variables and arrays

   printf("Enter number of elements in array\n");              // Prompt user to enter the number of elements
   scanf("%d", &n);                                            // Read the number of elements

   printf("Enter %d integers\n", n);                           // Prompt user to enter the integers
   for (c = 0; c < n; c++) {
      scanf("%d", &a[c]);                                      // Read each integer and store it in array 'a'
   }

   for (c = 0; c < n; c++) {
      for (d = 0; d < count; d++) {
         if (a[c] == b[d])                                     // Check if the current element is already in array 'b'
            break;
      }
      if (d == count) {                                        // If the element is not in array 'b'
         b[count] = a[c];                                      // Add the element to array 'b'
         count++;                                              // Increment the count of unique elements
      }
   }

   printf("Array obtained after removing duplicate elements:\n");      // Print the result
   for (c = 0; c < count; c++) {
      printf("%d\n", b[c]);                                    // Print each unique element
   }

   return 0;
}
```

OUTPUT

```
Enter number of elements in array
7
Enter 7 integers
7
5
7
5
4
2
1
Array obtained after removing duplicate elements:
7
5
4
2
1

Process returned 0 (0x0)   execution time : 11.804 s
```

Q14. Write a program which will arrange the positive and negative numbers in one dimensional array in such a way that all negative numbers should come first and then all the positive numbers will come without changing the original sequence of numbers.

Source Code

```
#include<stdio.h>

int main() {
   int pos[20], count = 0, neg[20], index = 0;                    // Declare integer arrays and variables
   int i, j, a[20];                                               // Declare integer variables and array
   printf("Enter any 6 Positive & Negative integers:\n");         // Prompt user to enter 6 integers
   for (i = 0; i < 6; i++) {
      scanf("%d", &a[i]);                                         // Read each integer and store it in array 'a'
   }

   for (i = 0; i < 6; i++) {
      if (a[i] >= 0) {
         pos[index] = a[i];                                       // Store positive integers in array 'pos'
         index++;
      } else {
         neg[count] = a[i];                                       // Store negative integers in array 'neg'
         count++;
      }
   }

   for (i = 0; i < index; i++) {
      a[i] = pos[i];                                              // Copy positive integers back to array 'a'
   }

   for (i = index, j = 0; i < 6; i++) {
      a[i] = neg[j];                                              // Copy negative integers back to array 'a'
      j++;
   }

   printf("Array after separating positive and negative integers:\n");   // Print the result
   for (i = 0; i < 6; i++) {
      printf("%d ", a[i]);                                        // Print each integer in the array
   }
   printf("\n");                                                  // Print a newline character

   return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\WR36D1~1.exe"
Enter any 6 Positive & Negative integers:
-8
-2
4
0
1
-2
Array after separating positive and negative integers:
4 0 1 -8 -2 -2

Process returned 0 (0x0)   execution time : 16.893 s
Press any key to continue.
```

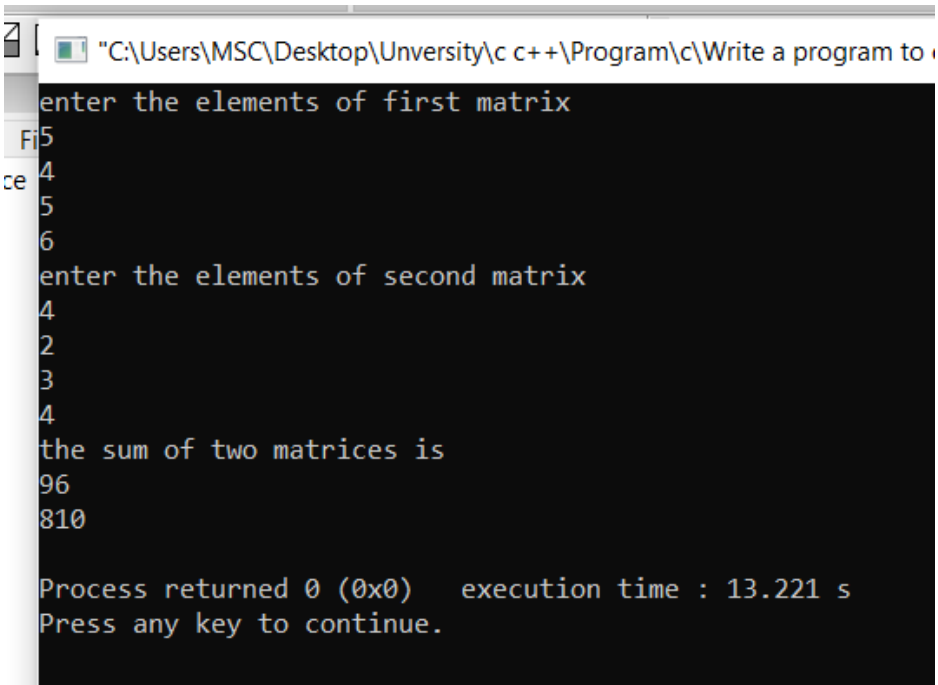
Q15. Write a program to compute Addition, Multiplication, Transpose on 2D array.

SOURCE CODE

ADDITION

```
#include<stdio.h>

int main() {
   int a[2][2], b[2][2], s[2][2], i, j;                    // Declare integer arrays and variables

   printf("Enter the elements of the first matrix\n");     // Prompt user to enter elements of the first matrix
   for (i = 0; i <= 1; i++) {
      for (j = 0; j <= 1; j++) {
         scanf("%d", &a[i][j]);                             // Read each element and store it in array 'a'
      }
   }

   printf("Enter the elements of the second matrix\n");    // Prompt user to enter elements of the second matrix
   for (i = 0; i <= 1; i++) {
      for (j = 0; j <= 1; j++) {
         scanf("%d", &b[i][j]);                             // Read each element and store it in array 'b'
      }
   }

   for (i = 0; i <= 1; i++) {
      for (j = 0; j <= 1; j++) {
         s[i][j] = a[i][j] + b[i][j];                       // Calculate the sum of corresponding elements and store it in array 's'
      }
   }

   printf("The sum of the two matrices is:\n");            // Print the result
   for (i = 0; i <= 1; i++) {
      for (j = 0; j <= 1; j++) {
         printf("%d ", s[i][j]);                            // Print each element of the sum matrix
      }
      printf("\n");                                         // Print a newline character after each row
   }

   return 0;
}
```

OUTPUT



SOURCE CODE

MULTIPLICATION

```
#include<stdio.h>

int main() {
   int a[10][10], b[10][10], s[10][10];                    // Declare integer arrays for matrices
   int m, n, l, p, i, j, k;                                // Declare integer variables
    printf("Enter the number of rows of the first matrix (<=10): ");   // Prompt user to enter the dimensions of the first matrix
   scanf("%d", &m);
   printf("Enter the number of columns of the first matrix (<=10): ");
   scanf("%d", &n);
   printf("Enter the number of rows of the second matrix (<=10): ");    // Prompt user to enter the dimensions of the second matrix
   scanf("%d", &l);
   printf("Enter the number of columns of the second matrix (<=10): ");
   scanf("%d", &p);
  printf("Enter the elements of the first matrix:\n");                    // Read elements of the first matrix
   for (i = 0; i < m; i++) {
      for (j = 0; j < n; j++) {
         scanf("%d", &a[i][j]);
      } }}
   printf("Enter the elements of the second matrix:\n");                    // Read elements of the second matrix
   for (i = 0; i < l; i++) {
      for (j = 0; j < p; j++) {
         scanf("%d", &b[i][j]);
      } }
   if (n != l) {                                                  // Check if matrix multiplication is possible
      printf("Multiplication not possible\n");
   } else {
            for (i = 0; i < m; i++) {                             // Perform matrix multiplication
         for (j = 0; j < p; j++) {
            s[i][j] = 0;
            for (k = 0; k < n; k++) {
               s[i][j] += a[i][k] * b[k][j];
            } }  }
            printf("The result of matrix multiplication is:\n");            // Print the result of matrix multiplication
      for (i = 0; i < m; i++) {
         for (j = 0; j < p; j++) {
            printf("%d ", s[i][j]);
         }
         printf("\n");
      }     }
   return 0;   }
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\Write a program to compute Multiplication on  2[
Enter the number of rows of the first matrix (<=10): 2
Enter the number of columns of the first matrix (<=10): 2
Enter the number of rows of the second matrix (<=10): 2
Enter the number of columns of the second matrix (<=10): 2
Enter the elements of the first matrix:
5
4
5
6
Enter the elements of the second matrix:
5
4
7
8
The result of matrix multiplication is:
53 52
67 68

Process returned 0 (0x0)   execution time : 11.406 s
Press any key to continue.
```

SOURCE CODE

TRANSPOSE

```
#include<stdio.h>

int main() {
   int a[3][3], b[3][3], i, j, m, n;                    // Declare integer arrays and variables

                                                        // Prompt user to enter the number of rows and columns of the matrix
   printf("Enter the number of rows of the matrix: ");
   scanf("%d", &n);
   printf("Enter the number of columns of the matrix: ");
   scanf("%d", &m);

                                                                // Read elements of the first matrix
   printf("Enter the elements of the first matrix:\n");
   for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++) {
         scanf("%d", &a[i][j]);
         b[j][i] = a[i][j];                                // Transpose the matrix while reading
      }
   }

      printf("Transpose of the matrix is:\n");                      // Print the transpose of the matrix
   for (i = 0; i < m; i++) {
      for (j = 0; j < n; j++) {
         printf("%d ", b[i][j]);
      }
      printf("\n");
   }

   return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\Write a program to compute
Enter the number of rows of the matrix: 3
Enter the number of columns of the matrix: 3
Enter the elements of the first matrix:
5
4
6
2
4
5
9
8
7
Transpose of the matrix is:
5 2 9
4 4 8
6 5 7

Process returned 0 (0x0)   execution time : 19.227 s
```

Q16. Implement a program which uses multiple files for holding multiple functions which are compiled separately, linked together and called by main(). Use static and extern variables in these files.

SOURCE CODE

MAIN.C

```c
#include <stdio.h>
#include "file1.h"                          // Include the header file

extern int externVar;                       // Declare the extern variable

int main() {
   printf("Initial value of externVar: %d\n", externVar);          // Print the initial value of externVar
   incrementExternVar();                                           // Call the function to increment externVar
   printf("Value of externVar after increment: %d\n", externVar);  // Print the value of externVar after increment
   printStaticVar();                                               // Call the function to print and increment the static variable
   return 0;
}
```

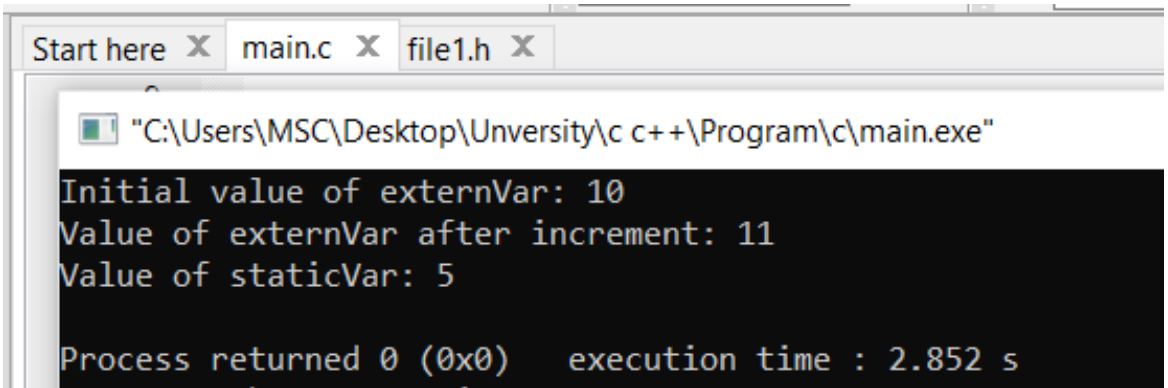
File1.h

```c
#ifndef FILE1_H                             // Prevent multiple inclusions of the header file
#define FILE1_H

extern int externVar;                       // Declare the extern variable

void incrementExternVar();                  // Declare the function to increment the extern variable
void printStaticVar();                      // Declare the function to print and increment the static variable

#endif
```

File1.c

```c
#include <stdio.h>
#include "file1.h"                          // Include the header file

int externVar = 10;                         // Define the extern variable

void incrementExternVar() {                 // Function to increment the extern variable
   externVar++;
}

void printStaticVar() {                     // Function to print and increment a static variable
   static int staticVar = 5;                // Define a static variable
   printf("Value of staticVar: %d\n", staticVar);
   staticVar++;
}
```

OUTPUT



LAB EXERCISES

Week 5

Q17. Implement a function which receives a pointer to a student struct and sets the value of its fields.
Q18. Write a program that takes five arguments on command line, opens a file and writes one argument per line in that file and closes the file
Q19. Write a program which creates Student (struct) objects using malloc and stores their pointers in an array. It must free the objects after printing their contents.
Q20. Write a function char* stuff (char *s1, char* s2, int sp, int rp) to suff string s2 in string s1 at position sp, replacing rp number of characters (rp may be zer0).

In this week

Macros in C are a powerful feature provided by the preprocessor, which allows you to define constants, functions, and code snippets that can be reused throughout your program. They are defined using the *#define* directive.

Types of Macros

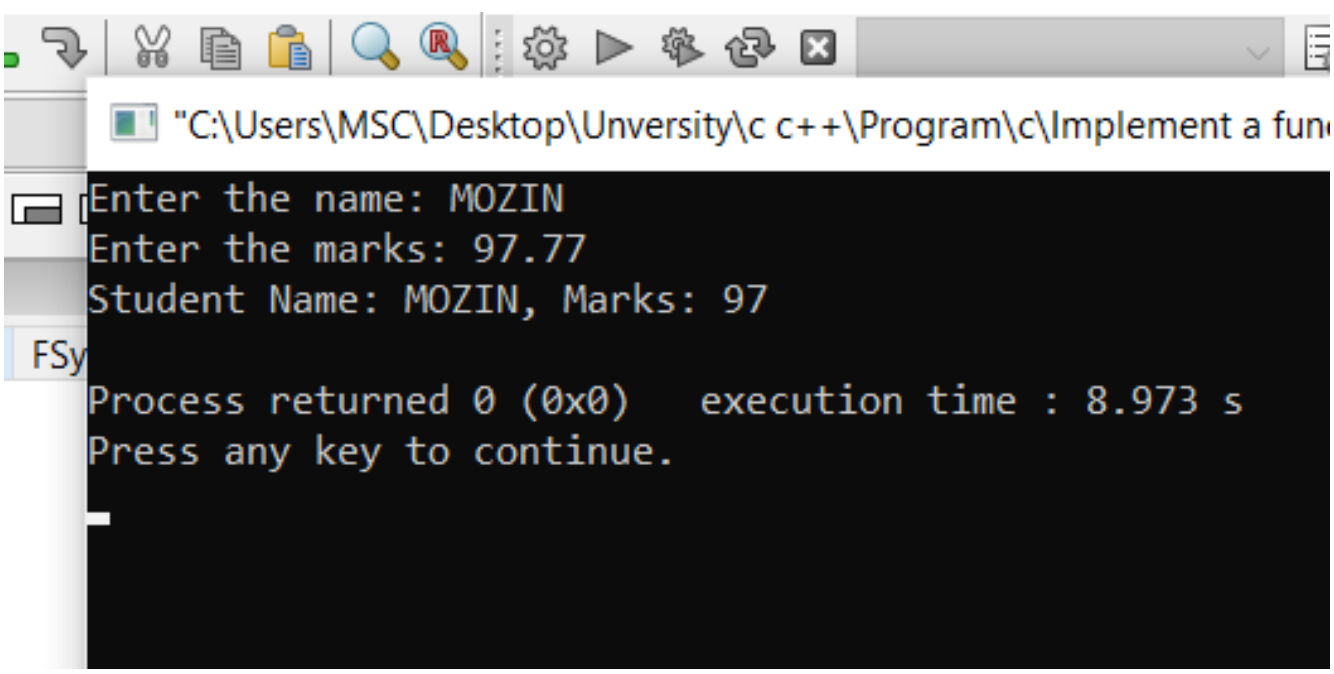
1. Object-like Macros
2. Function-like Macros

Q17. Implement a function which receives a pointer to a student struct and sets the value of its fields.

SOURCE CODE

```
#include<stdio.h>                                   // Include standard input-output library
struct student {                                    // Define a structure to store student information
   char name[20];
   int marks;
};
void print(struct student *ptr);                    // Function declaration

int main() {
   struct student s1;                       // Declare a variable of type struct student
   print(&s1);                                      // Call the function to input student details
   printf("Student Name: %s, Marks: %d\n", s1.name, s1.marks);      // Print the student details
   return 0;
}

                                        // Function to input student details
void print(struct student *ptr) {
   printf("Enter the name: ");              // Prompt user to enter the name
   scanf("%s", ptr->name);                  // Read the name and store it in the structure
   printf("Enter the marks: ");              // Prompt user to enter the marks
   scanf("%d", &ptr->marks);                // Read the marks and store it in the structure
}
```

OUTPUT



Q18. Write a program that takes five arguments on command line, opens a file and writes one argument per line in that file and closes the file.

SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>                          // Include standard library for exit macros

int main() {
                                              // Open a file for writing
   char *path = "output.txt";
   FILE *file = fopen(path, "w");
   if (!file) {
      perror(path);                          // Print error message if file cannot be opened
      return EXIT_FAILURE;                   // Return failure status
   }

                                             // Write "Hello Kashmir University" to the file
   fprintf(file, "Hello Kashmir University\n");

                                              // Close the file
   if (fclose(file)) {
      perror(path);                          // Print error message if file cannot be closed
      return EXIT_FAILURE;                   // Return failure status
   }

   printf("Message written to %s successfully.\n", path);          // Print success message
   return EXIT_SUCCESS;                                            // Return success status
}
```

OUTPUT

```
#include <stdlib.h>  // Include standard library for exit macros
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\Write a program that takes five arguments on command line, opens a file
Message written to output.txt successfully.

Process returned 0 (0x0)   execution time : 1.237 s
Press any key to continue.
```

OUTPUT.TXT FILE

output.txt - Notepad

File Edit Format View Help

Hello Kashmir University

Q19. Write a program which creates Student (struct) objects using malloc and stores their pointers in an array. It must free the objects after printing their contents.

SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define RECORDS 6                    // Define the number of student records

struct student {
   char name[25];
};

int main() {
   struct student *bonds[RECORDS];          // Array of pointers to struct student
   int index = 0;
   int count = 0;

                                            // Allocate memory for each student and initialize their names
   while (index < RECORDS) {
      bonds[index] = (struct student *)malloc(sizeof(struct student));
      if (bonds[index] == NULL) {
         fprintf(stderr, "Memory allocation failed\n");
         return EXIT_FAILURE;
      }
      strcpy(bonds[index]->name, "MRF");
      index++;
   }

                                             // Print the names of the students
   while (count < index) {
      printf("%s\n", bonds[count]->name);
      count++;
   }

                                                  // Free the allocated memory
   for (index = 0; index < RECORDS; index++) {
      free(bonds[index]);
   }

   return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\Write a program which creat
MRF
MRF
MRF
MRF
MRF
MRF

Process returned 0 (0x0)   execution time : 1.420 s
Press any key to continue.
```

Q20. Write a function char* stuff (char *s1, char* s2, int sp, int rp) to suff string s2 in string s1 at position sp, replacing rp number of characters (rp may be zer0).

SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
                                                    // Function to replace part of a string with another string

char *stuff(char *s1, char *s2, int sp, int rp) {
   int i = 0;
   while ((rp != 0) && (s1[sp] != '\0')) {
      s1[sp] = s2[i];
      sp++;
      i++;
      rp--;
   }
   return s1;
}

                                  /* WORKING PROGRAM */
int main() {
char t1[] = "helicopter";                  // Original string
   char t2[] = "XYZ";                       // String to insert
   printf("Old String is %s \n Insert String is %s \n new string is ", t1,t2);
   char *t3;

   t3 = stuff(t1, t2, 3, 4);               // Call the function to replace part of t1 with t2
   puts(t3);                               // Print the modified string

   return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\struct q20.exe"
Old String is helicopter
 Insert String is XYZ
 new string is helXYZ

Process returned 0 (0x0)   execution time : 1.605 s
Press any key to continue.
```

LAB EXERCISES

Week 6

Q21. Write a program to input name, address and telephone number of 'n' persons (n<=20). Sort according to the name as primary key and address as the secondary key. Print the sorted telephone directory.

Q22. Write a program to find the number of words in a sentence.

Q23. Write a program to concatenate two strings without using the inbuilt function.

Q24. Write a program to check if two strings are same or not.

Q25. Write a program to check whether a string is a palindrome or not.

Q26. Write a program to find the number of vowels and consonants in asentence.

In this week

A **FUNCTION** is a block of code that performs a specific task. Functions help in organizing code, making it reusable, and improving readability. In C++, functions can be predefined (standard library functions) or user-defined.

Q21. Write a program to input name, address and telephone number of 'n' persons (n<=20). Sort according to the name as primary key and address as the secondary key. Print the sorted telephone directory.

SOURCE CODE

```
#include <stdio.h>
#include <string.h>
#define SIZE 3                  // Define the number of records
struct jb {                     // Define a structure to store information
   char name[25];
   char address[50];
   int telephone;
};
int main() {
   struct jb bonds[SIZE];          // Array of structures
   struct jb temp;                 // Temporary variable for swapping
   int i = 0, x, a, b;
   int phno;
   char getName[25];
   char getAddress[50];
   while (i < SIZE) {              // Getting user data
      phno = 0;
      printf("Enter your Name: ");
      scanf("%s", getName);                          // Read the name
      printf("Enter your Address: ");
      scanf("%s", getAddress);                       // Read the address
      strcpy(bonds[i].name, getName);                // Copy name to structure
      strcpy(bonds[i].address, getAddress);          // Copy address to structure
      printf("Enter your phone: ");
      scanf("%d", &phno);                            // Read the phone number
      bonds[i].telephone = phno;                     // Store phone number in structure
      i++;
   }                                                 // Initial array display
   printf("Initial array display:\n");
   for (x = 0; x < SIZE; x++) {
      printf("%s\n", bonds[x].name);                 // Print each name
   }
   for (a = 0; a < SIZE - 1; a++) {                  // Sorting on the basis of name
      for (b = a + 1; b < SIZE; b++) {
         x = 0;
         while (bonds[a].name[x]) {
            if ((bonds[a].name[x]) > (bonds[b].name[x])) {
               temp = bonds[a];
               bonds[a] = bonds[b];
               bonds[b] = temp;
               break;
            } else if ((bonds[a].name[x]) < (bonds[b].name[x])) {
               break;
            } else {
               x++;
            }        }      } }
      printf("Modified array:\n");                   // Modified array display
   for (x = 0; x < SIZE; x++) {
      printf("Name: %s\n", bonds[x].name);           // Print each name
      printf("Address: %s\n", bonds[x].address);     // Print each address
      printf("Telephone: %d\n", bonds[x].telephone);          // Print each phone number
   }
   return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\q21 n20.exe"
Enter your Name: Mozin
Enter your Address: Beehama
Enter your phone: 9622424392
Enter your Name: Shariq
Enter your Address: Kangan
Enter your phone: 6005248524
Enter your Name: Bisma
Enter your Address: Lar
Enter your phone: 7051587695
Initial array display:
Mozin
Shariq
Bisma
Modified array:
Name: Bisma
Address: Lar
Telephone: -1538346897
Name: Mozin
Address: Beehama
Telephone: 1032489800
Name: Shariq
Address: Kangan
Telephone: 1710281228

Process returned 0 (0x0)   execution time : 57.217 s
Press any key to continue.
```

Q22. Write a program to find the number of words in a sentence.

SOURCE CODE

```
#include<stdio.h>
#include<string.h>

int main() {
   char strs[100];                // Array to store the input sentence
   int countw = 0;                // Variable to count the number of words
   int i;                         // Loop variable
                                  // Prompt the user to input a sentence
   printf("input sentence : ");
   gets(strs);                    // Read the input sentence

   int len = strlen(strs);        // Calculate the length of the input sentence

                                  // Loop through each character in the sentence
   for(i = 0; i < len; i++) {
                                  // Check if the current character is a space
      if(strs[i] == ' ') {
         countw++;                // Increment the word count
      }
   }
                                  // Print the total number of words in the sentence
   printf("Total number of words in the sentence is %d", countw + 1);

   return 0;
}
```

OUTPUT

```
input sentence : I AM HERE
Total number of words in the sentence is 3
Process returned 0 (0x0)   execution time : 36.546 s
Press any key to continue.
```

Q23. Write a program to concatenate two strings without using the inbuilt function.

SOURCE CODE

```
include <stdio.h>

int main() {
   char s1[100], s2[100];            // Arrays to store the input strings
   int i, j;           // Loop variables
   printf("Enter first string: ");                  // Prompt the user to enter the first string
   scanf("%s", s1);
   printf("Enter second string: ");              // Prompt the user to enter the second string
   scanf("%s", s2);
      for(i = 0; s1[i] != '\0'; ++i);     // Calculate the length of string s1 and store it in i
      for(j = 0; s2[j] != '\0'; ++j, ++i) {       // Concatenate s2 to the end of s1
      s1[i] = s2[j];
   }
   s1[i] = '\0';                // Null-terminate the concatenated string
   printf("After concatenation: %s", s1);    // Print the concatenated string
   return 0;
}
```

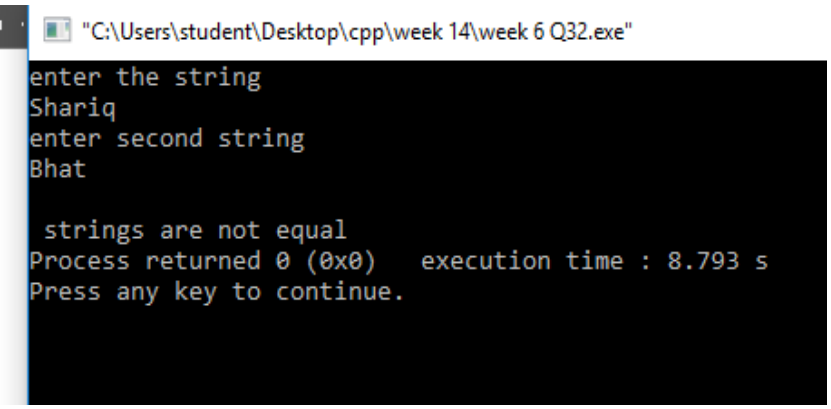
OUTPUT

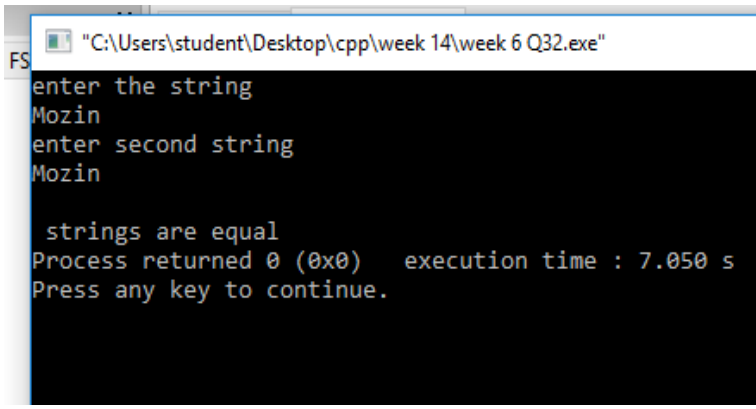
```
"C:\Users\student\Desktop\cpp\week 14\week 6 Q31.exe"
ter first string: MOZIN
ter second string: AHMAD
ter concatenation: MOZINAHMAD
ocess returned 0 (0x0)   execution time : 10.148
ess any key to continue.
```

Q24. Write a program to check if two strings are same or not.

```
#include<stdio.h>
#include<string.h>

int main() {
   char string1[50];              // Array to store the first string
   char string2[50];              // Array to store the second string
   int i, j, flag;                              // Variables for indexing and flag
   printf("enter the string\n");                // Prompt the user to enter the first string
   gets(string1);
   printf("enter second string\n");      // Prompt the user to enter the second string
   gets(string2);
   i = 0;                                       // Initialize indices and flag
   j = 0;
   flag = 0;

      while (string1[i] != '\0')                // Calculate the length of the first string
      i++;

      while (string2[j] != '\0')  // Calculate the length of the second string
      j++;

      if (i != j)                               // If lengths are not equal, set flag to 1
      flag = 1;
   else {

      i = 0;              // Reset indices
      j = 0;
                          // Compare characters of both strings
      while ((string1[i] != '\0') && (string2[j] != '\0')) {
         if (string1[i] != string2[j]) {
            flag = 1; // Set flag if characters are not equal
            break;
         }
         j++;
         i++;
      }
```

OUTPUT

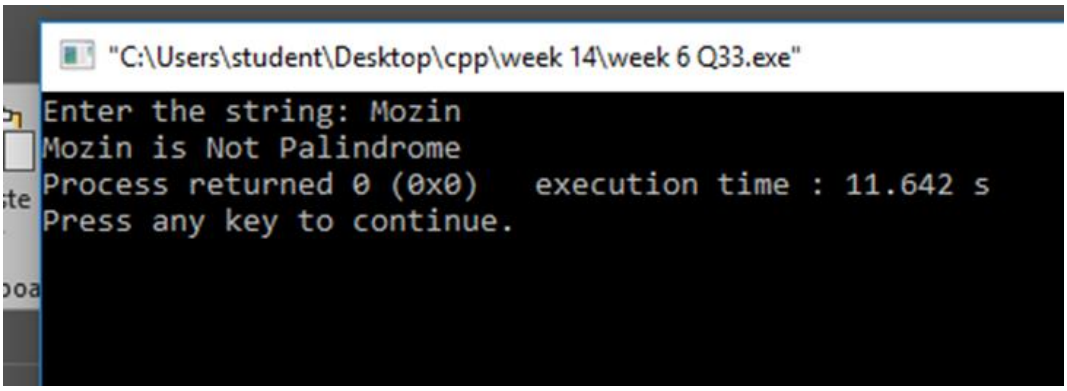




Q25. Write a program to check whether a string is a palindrome or not

```
#include<stdio.h>
#include<string.h>
// Function declaration to check if a string is a palindrome
void isPalindrome(char str[]);
int main() {
   char s1[100];                          // Array to store the input string
   // Prompt the user to enter a string
   printf("Enter the string: ");
   scanf("%s", s1);
   // Call the function to check if the string is a palindrome
   isPalindrome(s1);
   return 0;
}
// A function to check if a string str is a palindrome
void isPalindrome(char str[]) {
                          // Start from leftmost and rightmost corners of str
   int l = 0;
   int h = strlen(str) - 1;
   // Keep comparing characters while they are the same
   while (h > l) {
      if (str[l++] != str[h--]) {
                          // If characters do not match, print that the string is not a palindrome
         printf("%s is Not Palindrome", str);
         return;
      }
   }
   // If all characters match, print that the string is a palindrome
   printf("%s is palindrome", str);
}
```

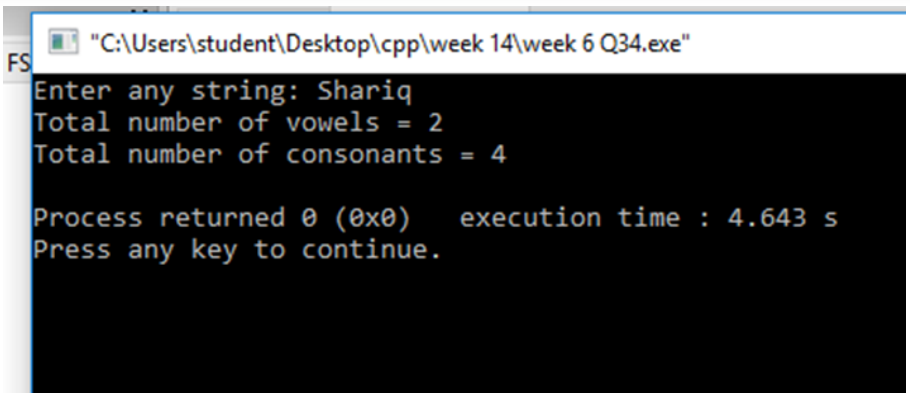
OUTPUT



Q26. Write a program to find the number of vowels and consonants in a sentence.

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 100

int main() {
   char str[MAX_SIZE];                    // Array to store the input string
   int i, len, vowel, consonant;          // Variables for indexing, length, vowel count, and consonant count
   // Input string from user
   printf("Enter any string: ");
   gets(str);                        // Read the input string
   vowel = 0;                        // Initialize vowel count to 0
   consonant = 0;    // Initialize consonant count to 0
   len = strlen(str);  // Calculate the length of the input string

            // Loop through each character in the string
   for(i = 0; i < len; i++) {
            // Check if the current character is an alphabet
     if((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z')) {
            // If the current character is a vowel
        if(str[i] == 'a' || str[i] == 'e' || str[i] == 'i' ||
          str[i] == 'o' || str[i] == 'u' ||
          str[i] == 'A' || str[i] == 'E' || str[i] == 'I' ||
          str[i] == 'O' || str[i] == 'U') {
           vowel++;                  // Increment vowel count
        } else {
           consonant++;                    // Increment consonant count
        }
     }
   }
   // Print the total number of vowels and consonants
   printf("Total number of vowels = %d\n", vowel);
   printf("Total number of consonants = %d\n", consonant);

   return 0;
}
```

OUTPUT



LAB EXERCISES

Week 7

Q27. Write a program that reverse the contents of a string.

Q28. Write a program to demonstrate the array indexing using pointers..

Q29. Write a program to pass a pointer to a structure as a parameter to a function and return back a pointer to structure to the calling function after modifying the members of structure.

Q30. Write a program to check if two strings are same or not.

In this week

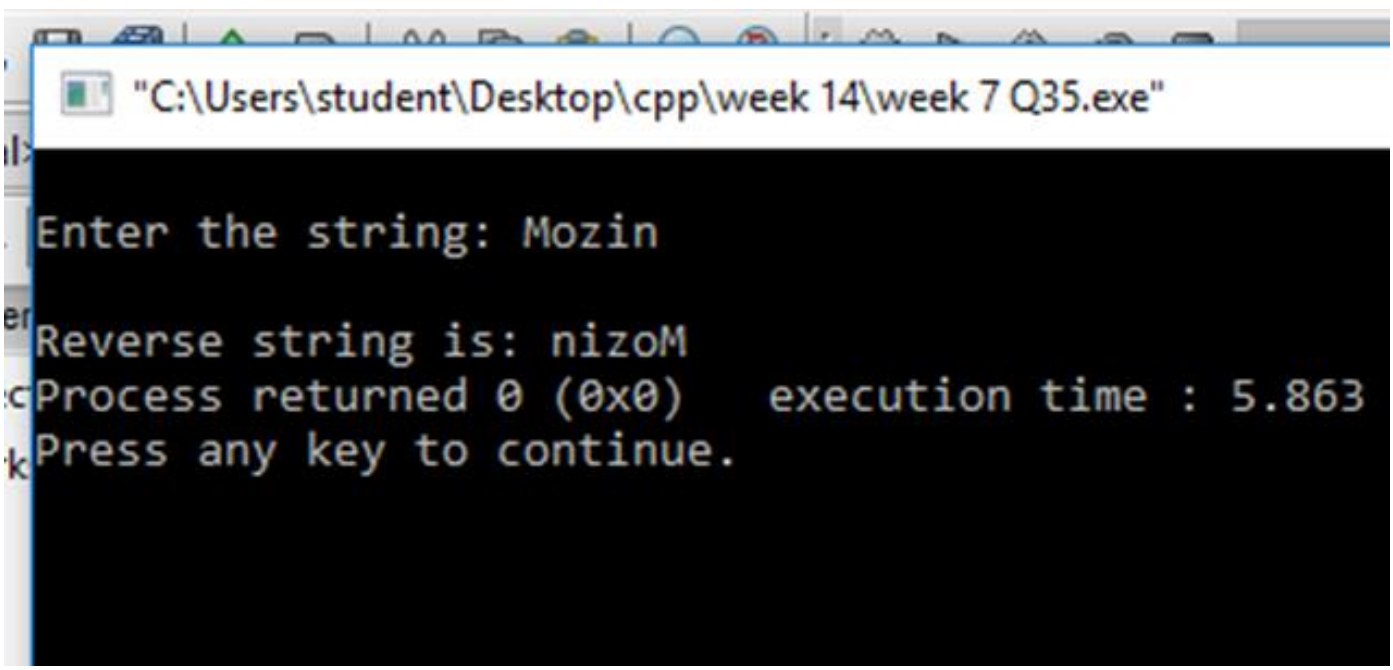
Array indexing refers to accessing individual elements of an array using their position (index) within the array. In C++, arrays are zero-indexed, meaning the first element is at index 0, the second element is at index 1, and so on.

Q27. Write a program that reverse the contents of a string..

SOURCE CODE

```
#include<stdio.h>
#include<string.h>
int main() {
   char str[100], temp;                                  // Array to store the input string and a temporary variable for swapping
   int i, j = 0;                                 // Variables for indexing
 printf("\nEnter the string: ");      // Prompt the user to enter a string
   gets(str);                                        // Read the input string
   i = 0;                                            // Initialize the starting index
   j = strlen(str) - 1;                  // Initialize the ending index
   // Loop to reverse the string
   while (i < j) {
                                 // Swap the characters at positions i and j
      temp = str[i];
      str[i] = str[j];
      str[j] = temp;
      i++; // Move the indices towards the center
      j--;
   }
      printf("\nReverse string is: %s", str);      // Print the reversed string
   return 0;
}
```

OUTPUT

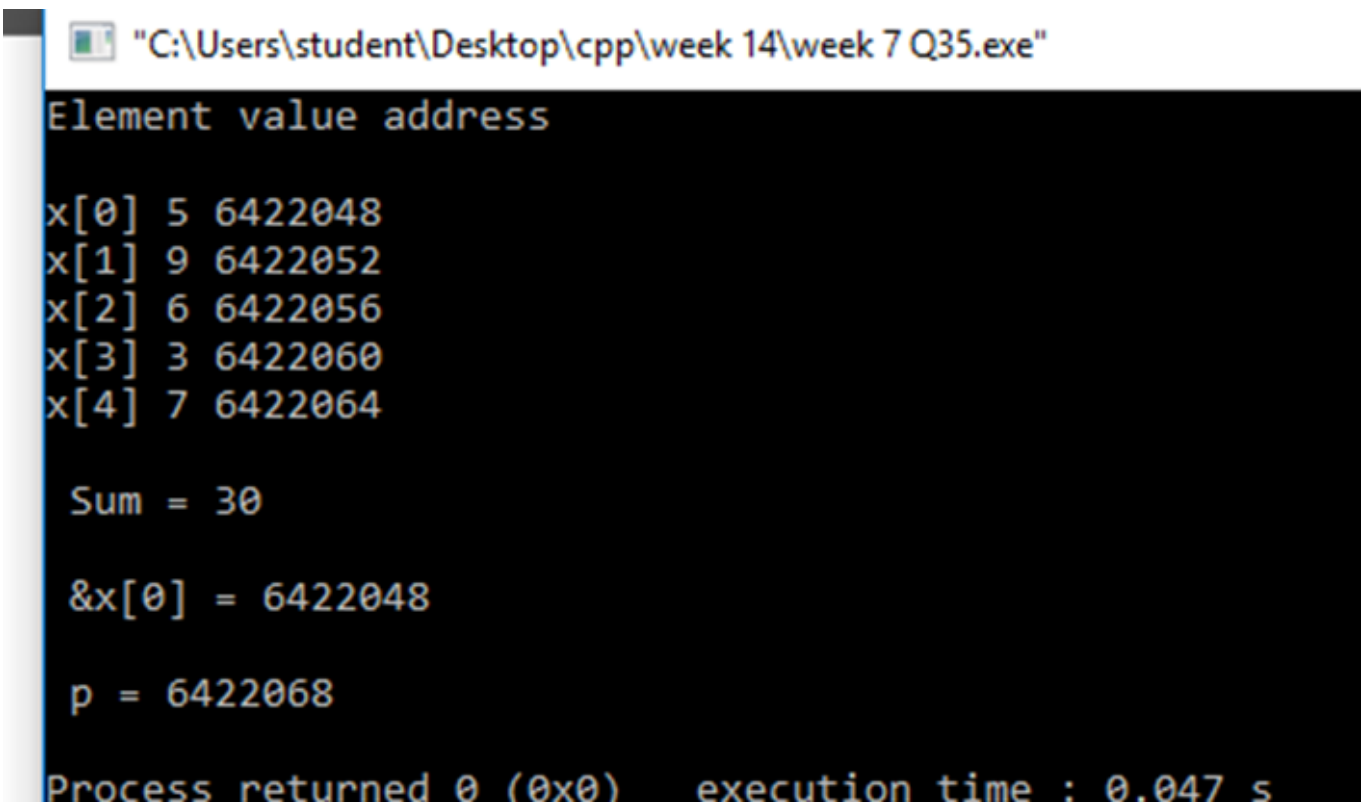


Q28. Write a program to demonstrate the array indexing using pointers..

SOURCE CODE

```
#include<stdio.h>
int main() {
   int *p, sum = 0, i;              // Pointer to int, sum variable, and loop variable
   int x[5] = {5, 9, 6, 3, 7};      // Array of integers
   i = 0;
                                     // Initializing pointer p with the base address of array x
   p = x;
    printf("Element value address\n\n");

   while(i < 5) {                    // Loop through the array using the pointer
      printf("x[%d] %d %u\n", i, *p, p);        // Print the element, its value, and its address
      // Add the value pointed to by p to sum
      sum = sum + *p; // Accessing array element
      i++;
      p++; // Move the pointer to the next element
   }
   // Print the sum of the array elements
   printf("\n Sum = %d\n", sum);
   // Print the address of the first element of the array
   printf("\n &x[0] = %u\n", &x[0]);
            // Print the current value of the pointer p (which is now beyond the last element)
   printf("\n p = %u\n", p);
   return 0;
}
```

OUTPUT



Q29. Write a program to pass a pointer to a structure as a parameter to a function and return back a pointer to structure to the calling function after modifying the members of structure.

SOURCE CODE

```
#include<stdio.h>
struct student {                    // Structure definition for student
   char name[20];                    // Name of the student
   int marks;                        // Marks of the student
};
              // Function declaration to modify student details
struct student* print(struct student *ptr);
int main() {
   struct student s1, *s2;        // Declare a student structure and a pointer to student
      printf("enter the name: ");           // Prompt the user to enter the name
   scanf("%s", s1.name);
      printf("enter the marks: ");                        // Prompt the user to enter the marks
   scanf("%d", &s1.marks);
    s2 = print(&s1);               // Call the print function to modify student details
    printf("contents after modifying\n");  // Display the modified contents
   printf("%s %d", s2->name, s2->marks);
   return 0;
}
              // Function definition to modify student details
struct student* print(struct student *ptr) {
               // Prompt the user to enter the name
   printf("enter the name: ");
   scanf("%s", ptr->name);
                 // Prompt the user to enter the marks
   printf("enter the marks: ");
   scanf("%d", &ptr->marks);
   return ptr;
}
```

OUTPUT

```
\Users\student\Desktop\cpp\week 14\week 7 Q37.exe"
the name: Shariq
the marks: 97
the name: Mozin
the marks: 97
nts after modifying
97
ss returned 0 (0x0)   execution time : 10.715 s
any key to continue.
```

LAB EXERCISES

Week 8

Q30. . Write a program to demonstrate the use of pointer to a pointer
Q31. Write a program to demonstrate the use of pointer to a function
Q32. Write a program to demonstrate the swapping the fields of two structures using pointers
Q33. Write a program in C++ to define class complex having two data members viz real and imaginary part
Q34. Write a program in C++ to define class Person having multiple data members for storing the different details of person e.g. name, age, address, height.

In this week

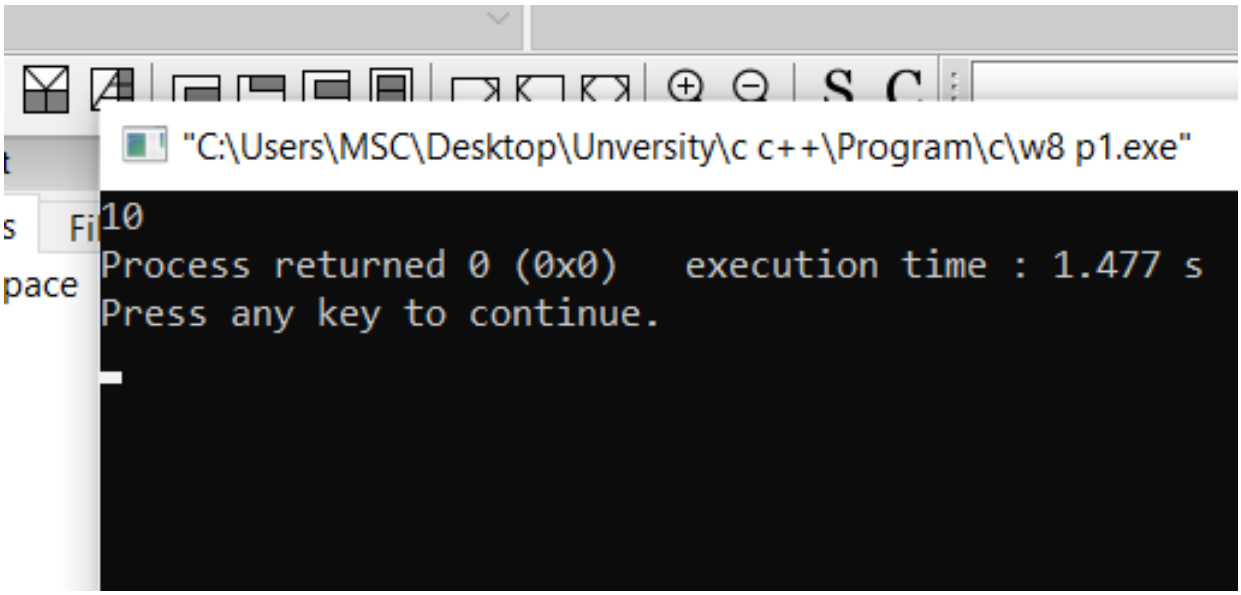
STRUCTURE in C++ is a user-defined data type that allows grouping variables of different types under a single name. Structures are similar to classes but have some differences in terms of access control and default member access.

Q30. Write a program to demonstrate the use of pointer to a pointer

SOURCE CODE

```
#include<stdio.h>
int main()
{
int x, *p, **q;
x=10;
p=&x;
q=&p;
// print the value of x
printf("%d", **q);
return 0;
}
```

OUTPUT



Q31. Write a program to demonstrate the use of pointer to a function

SOURCE CODE

```
#include <stdio.h>
#include <string.h>
void check(char *a, char *b, int (*cmp)(const char *, const char *));  // Function prototype declaration
int main() {
   char s1[80], s2[80];
   int (*p)(const char *, const char *);
   p = strcmp;                          // Assign the strcmp function to the function pointer
  printf("Enter first string: ");       // Get input strings from the user
   gets(s1);
   printf("Enter second string: ");
   gets(s2);
   check(s1, s2, p);                     // Call the check function to compare the strings
   return 0;
}
            // Function to check if two strings are equal using a function pointer
void check(char *a, char *b, int (*cmp)(const char *, const char *)) {
   printf("Testing for equality\n");
   if (!(*cmp)(a, b)) {
      printf("Strings are Equal\n");
   } else {
      printf("Strings are Not Equal\n");
   }
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\w8 p1.cp2.exe"
Enter first string: mozin
Enter second string: shariq
Testing for equality
Strings are Not Equal
```

Q32. Write a program to demonstrate the swapping the fields of two structures using pointers

SOURCE CODE

```
#include <stdio.h>
#include <string.h>
struct student {                          // Define a structure to store student information
   char name[25];
   int marks;
};
void swap(struct student *a, struct student *b) {          // Function to swap the fields of two structures
   struct student temp;
      strcpy(temp.name, a->name);                  // Swap the name fields
   strcpy(a->name, b->name);
   strcpy(b->name, temp.name);
   temp.marks = a->marks;                            // Swap the marks fields
   a->marks = b->marks;
   b->marks = temp.marks;
}
int main() {
   struct student s1, s2;
   strcpy(s1.name, "Alice");                  // Initialize the first student
   s1.marks = 85;
   strcpy(s2.name, "Bob");                           // Initialize the second student
   s2.marks = 90;
    printf("Before swapping:\n");                  // Print initial values
   printf("Student 1: %s, Marks: %d\n", s1.name, s1.marks);
   printf("Student 2: %s, Marks: %d\n", s2.name, s2.marks);
    swap(&s1, &s2);                                // Swap the fields of the two students
      printf("After swapping:\n");                          // Print values after swapping
   printf("Student 1: %s, Marks: %d\n", s1.name, s1.marks);
   printf("Student 2: %s, Marks: %d\n", s2.name, s2.marks);
   return 0;
}
```

OUTPUT

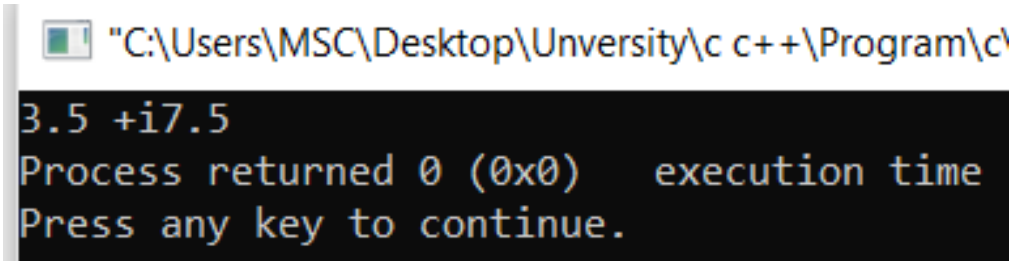
```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\w8 p1.cp3.exe"
Before swapping:
Student 1: Alice, Marks: 85
Student 2: Bob, Marks: 90
After swapping:
Student 1: Bob, Marks: 90
Student 2: Alice, Marks: 85

Process returned 0 (0x0)   execution time : 0.946 s
Press any key to continue.
```

Q33. Write a program in C++ to define class complex having two data members viz real and imaginary part

SOURCE CODE

```
#include <iostream>
using namespace std;
class complex {
private:
   float real;                          // Real part of the complex number
   float imaginary;                     // Imaginary part of the complex number
public:
   void getdata() {                     // Function to set the data members
      real = 3.5;
      imaginary = 7.5;
   }
      void putdata() {                  // Function to display the complex number
      cout << real << " +i" << imaginary;
   }
};
int main() {
   complex c;                           // Create an object of the complex class
   c.getdata();                         // Set the data members of the object
   c.putdata();                         // Display the complex number
   return 0;                            // Return 0 to indicate successful execution
}
```

OUTPUT



Q34. Write a program in C++ to define class Person having multiple data members for storing the different details of person e.g. name, age, address, height.

SOURCE CODE

```
#include <iostream>
using namespace std;

// Define a class to represent a person
class person {
private:
   char name[20];                    // Name of the person
   char address[20];                 // Address of the person
   int age;                           // Age of the person
   float height;                     // Height of the person

public:
                        // Function to get data from the user
   void getdata() {
      cout << "Enter name: " << endl;
      cin >> name;
      cout << "Enter address: " << endl;
      cin >> address;
      cout << "Enter age: " << endl;
      cin >> age;
      cout << "Enter height: " << endl;
      cin >> height;
   }

                       // Function to display the data
   void putdata() {
      cout << "Name is: " << name << endl;
      cout << "Address is: " << address << endl;
      cout << "Age is: " << age << endl;
      cout << "Height is: " << height << endl;
   }
};

int main() {
   person p;                  // Create an object of the person class
   p.getdata();                // Get data from the user
   p.putdata();               // Display the data
   return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\w8 p5.cp3.exe"
enter name
MOzin
enter address
Beehama
enter age
27
enter height
5.7
name is: MOzin
address is: Beehama
age is: 27
height is: 5.7
```

LAB EXERCISES

Week 9

Q35. Write a program to instantiate the objects of class person and class complex
Q36. Write a program to demonstrate the use of pointer to a function
Q37. Write a C++ program to demonstrate the use of scope resolution operator
Q38. Write a program in C++ which creates objects of Student class using default, overloaded and copy constructors.

In this week

A **POINTER** * is a variable that stores the memory address of another variable. Pointers are used for dynamic memory allocation, arrays, and functions.

A **POINTER TO POINTER** ** is a variable that stores the address of another pointer. This allows for multiple levels of indirection.

Q35. Write a program to instantiate the objects of class person and class complex

SOURCE CODE

```
#include <iostream>
using namespace std;
class complex {                    // Define the complex class
private:
   float real;
   float imaginary;
public:
      void getdata() {                        // Function to set data members
      real = 3.5;
      imaginary = 7.5;
   }
      void putdata() {                        // Function to display the complex number
      cout << real << " +i" << imaginary << endl;
   }         };
class person {                     // Define the person class
private:
   char name[20];
   char address[20];
   int age;
   float height;

public:
      void getdata() {                        // Function to get data from the user
      cout << "Enter name: ";
      cin >> name;
      cout << "Enter address: ";
      cin >> address;
      cout << "Enter age: ";
      cin >> age;
      cout << "Enter height: ";
      cin >> height;
   }
      void putdata() {                        // Function to display the data
      cout << "Name: " << name << endl;
      cout << "Address: " << address << endl;
      cout << "Age: " << age << endl;
      cout << "Height: " << height << endl;
   }                    };
int main() {                          // Instantiate an object of the complex class
   complex c;
   c.getdata();
   c.putdata();
   person p;                          // Instantiate an object of the person class
   p.getdata();
   p.putdata();

   return 0;
}
```

OUTPUT

```
"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\w9 p1.exe"
3.5 +i7.5
Enter name: Mozin
Enter address: Beehama
Enter age: 27
Enter height: 5.7
Name: Mozin
Address: Beehama
Age: 27
Height: 5.7
```

Q36. Write a C++ program to add member function that displays the contents of class person and class complex

SOURCE CODE

```
#include <iostream>
using namespace std;
class complex {                    // Define a class to represent complex numbers
private:
   float real;                     // Real part of the complex number
   float imaginary;                // Imaginary part of the complex number
public:
   void get_complex() {                    // Function to get the real and imaginary parts from the user
      cout << "Enter the real part: " << endl;
      cin >> real;
      cout << "Enter the imaginary part: " << endl;
      cin >> imaginary;
   }
   void show_complex() {                   // Function to display the complex number
      cout << real << " +i" << imaginary << endl;
   }        };
class person {                             // Define a class to represent a person
private:
   char name[20];                          // Name of the person
   char address[20];                       // Address of the person
   int age;                                // Age of the person
   float height;                           // Height of the person
public:
   void getdata() {                                   // Function to get data from the user
      cout << "Enter name: " << endl;
      cin >> name;
      cout << "Enter address: " << endl;
      cin >> address;
      cout << "Enter age: " << endl;
      cin >> age;
      cout << "Enter height: " << endl;
      cin >> height;
   }
   void putdata() {                                           // Function to display the data
      cout << "Name is: " << name << endl;
      cout << "Address is: " << address << endl;
      cout << "Age is: " << age << endl;
      cout << "Height is: " << height << endl;
   }        };
int main() {
   complex c1;                     // Create an object of the complex class
   c1.get_complex();               // Get the real and imaginary parts from the user
   c1.show_complex();              // Display the complex number
   person p1;                      // Create an object of the person class
   p1.getdata();                   // Get data from the user
   p1.putdata();                   // Display the data
   return 0;
}
```

OUTPUT

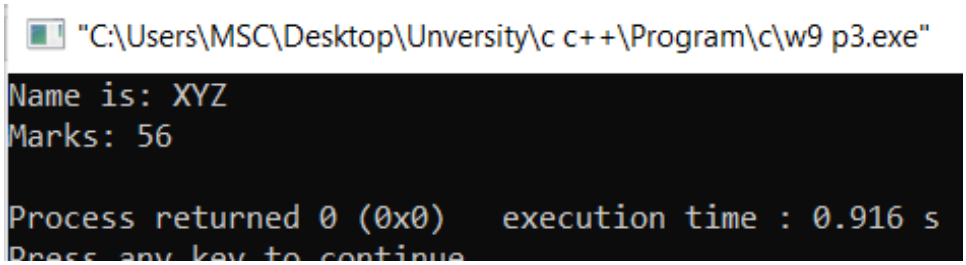
```
"C:\Users\MSC\Desktop\Unversity\c c++\Pro
Enter the real part:
7
Enter the imaginary part:
9
7 +i9
Enter name:
kashmir
Enter address:
J&K
Enter age:
57
Enter height:
500
Name is: kashmir
Address is: J&K
Age is: 57
Height is: 500
```

Q37. Write a C++ program to demonstrate the use of scope resolution operator

SOURCE CODE

```
#include <iostream>
#include <string.h>
using namespace std;
class student {                    // Define a class to represent a student
private:
   char name[20];                  // Name of the student
   int marks;                       // Marks of the student
public:
   void get(char *n, int a);                    // Function to set the name and marks of the student
    void put();                        // Function to display the name and marks of the student
};
void student::get(char *n, int a) {                    // Function to set the name and marks of the student
   strcpy(name, n);                            // Copy the name to the name field
   marks = a;                                  // Set the marks field
}
void student::put() {                         // Function to display the name and marks of the student
   cout << "Name is: " << name << endl;
   cout << "Marks: " << marks << endl;
}
int main() {
   student s1;                     // Create an object of the student class
   s1.get("XYZ", 56);                          // Set the name and marks of the student
   s1.put();                                   // Display the name and marks of the student
   return 0;                                   // Return 0 to indicate successful execution
}
```

OUTPUT

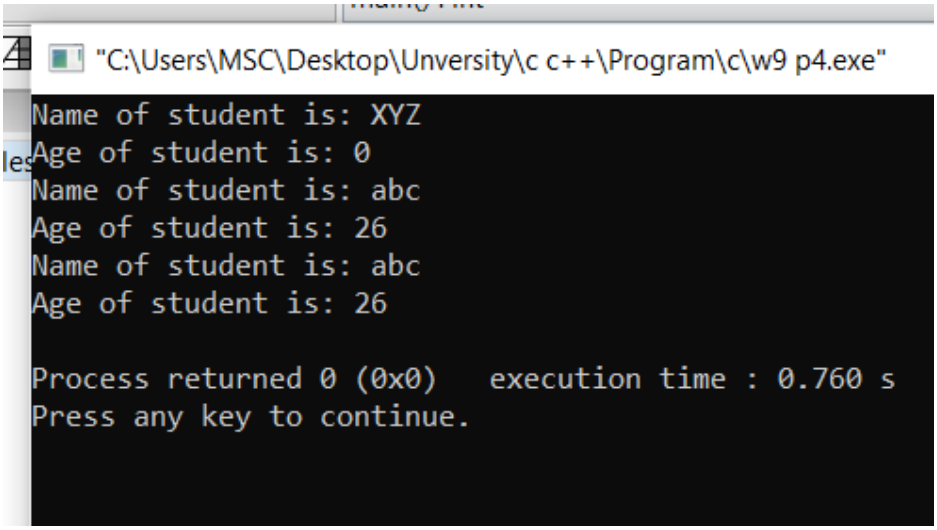


Q38. Write a program in C++ which creates objects of Student class using default, overloaded and copy constructors.

SOURCE CODE

```
#include <iostream>
using namespace std;
#include <string.h>
class student {                          // Define a class to represent a student
private:
   char name[20];                         // Name of the student
   int age;                               // Age of the student
public:
      student() {};                       // Default constructor
   student(char *n) {                       // Constructor with name parameter
      strcpy(name, n);
      age = 0;
   }
      student(char *n, int a) {                    // Constructor with name and age parameters
      strcpy(name, n);
      age = a;
   }
      student(student &s) {                        // Copy constructor
      strcpy(name, s.name);
      age = s.age;
   }
      void show();                         // Function to display the student's details
};
void student::show() {                     // Function to display the student's details
   cout << "Name of student is: " << name << endl;
   cout << "Age of student is: " << age << endl;
}
int main() {
   student s2("XYZ");                      // Create an object using the constructor with name parameter
   student s3("abc", 26);                   // Create an object using the constructor with name and age parameters
   student s4(s3);                         // Create an object using the copy constructor

   s2.show();                              // Display the details of s2
   s3.show();                              // Display the details of s3
   s4.show();                              // Display the details of s4

   return 0;
}
```

OUTPUT



LAB EXERCISES

Week 10

Q39. Write a program to demonstrate the use of different access specifiers.
Q40. Write a C++ program to demonstrate the use of inline, friend functions and this keyword.
Q41. Write a C++ program to show the use of destructors.
Q42. Write a program in C++ demonstrates the use of function overloading.
Q43. Write a C++ program to overload the '+' operator so that it can add two matrices.

In this week

Overloading is a feature in C++ where multiple functions can have the same name with different parameters. The correct function is selected based on the arguments passed.

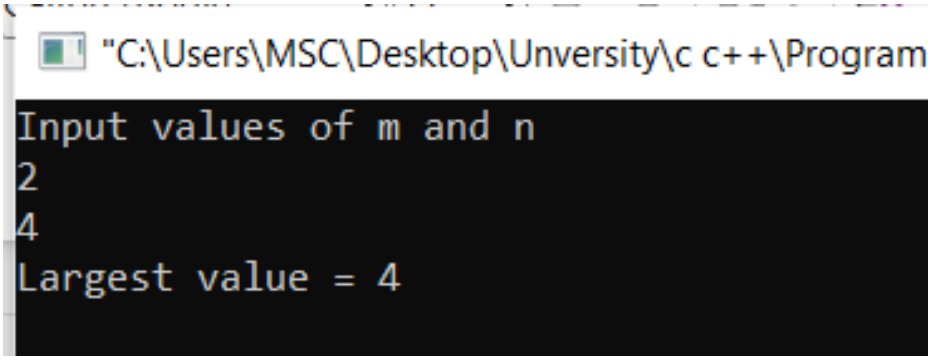
It refers to the ability to define multiple functions with the same name but different parameters within the same scope. This allows functions to handle

Q39. Write a program to demonstrate the use of different access specifiers

SOURCE CODE

```
#include <iostream>
using namespace std;

        // Define a class to represent a sample
class sample {
private:
   int m, n;                    // Private data members
   void display();              // Private member function

public:
   void input();                // Public member function to input values
   int largest();               // Public member function to find the largest value
};

        // Function to find the largest value between m and n
int sample::largest() {
   if (m >= n)
      return (m);
   else
      return (n);
}

        // Function to input values for m and n
void sample::input() {
   cout << "Input values of m and n" << "\n";
   cin >> m >> n;
}

        // Function to display the largest value
void sample::display() {
   cout << "Largest value = " << largest() << "\n";
}

int main() {
   sample A;                    // Create an object of the sample class
   int temp;
   A.input();                   // Input values for m and n
   temp = A.largest();          // Find the largest value
   cout << "Largest value = " << temp << "\n";
                                // A.display();          // Objects can't access private members
   return 0;
}
```

OUTPUT



Q40. Write a C++ program to demonstrate the use of inline, friend functions and this keyword.

SOURCE CODE

```
#include <iostream>
using namespace std;
class sample {                    // Define a class to represent a sample
private:
   int m;                 // Private data member for the first value
   int n;                  // Private data member for the second value

   void display();                // Private member function to display the values
public:
      friend int sum(sample x);                    // Friend function declaration to access private members
      void set_mn(int m, int n);                   // Public member function to set the values of m and n
     void put_mn();                                // Public member function to display the values of m and n
};
void sample::set_mn(int m, int n) {                // Function to set the values of m and n
   this->m = m;
   this->n = n;
}
inline void sample::put_mn() {                     // Inline function to display the values of m and n
   cout << "m: " << m << endl << "n: " << n << endl;
}
int sum(sample x) {                                // Friend function to calculate the sum of m and n
   return x.m + x.n;
}

int main() {
   sample s1, s2;                          // Create objects of the sample class

   // Set and display values for the first object
   s1.set_mn(7, 8);
   s1.put_mn();

   // Set and display values for the second object
   s2.set_mn(56, 8);
   s2.put_mn();

   // Calculate and display the sum of values in the first object
   cout << "Sum of m and n in s1: " << sum(s1) << endl;

   return 0;
}
```

Q41. Write a C++ program to show the use of destructors.

SOURCE CODE

```
#include <iostream>
#include <string.h>
using namespace std;
int count = 0;                    // Global variable to keep track of the number of objects
class student {                   // Define a class to represent a student
private:
   int roll;                      // Roll number of the student
   int age;                       // Age of the student
   char name[20];                 // Name of the student
public:
      student(int r, int a, char *n);        // Constructor to initialize the student object
   ~student();                                // Destructor to handle object destruction
      void get_details();                    // Function to display the details of the student
};
student::student(int r, int a, char *z) {                    // Constructor definition
   count++;                                                  // Increment the count of objects created
   cout << "No. of objects created " << count << endl;
   age = r;
   roll = a;
   strcpy(name, z);                                          // Copy the name to the name field
}
student::~student() {                                                // Destructor definition
   cout << "No. of objects destroyed " << count << endl;
   count--;                                                  // Decrement the count of objects destroyed
}
void student::get_details() {                                // Function to display the details of the student
   cout << "Name: " << name << endl;
   cout << "Age: " << age << endl;
   cout << "Roll: " << roll << endl;
}

int main() {
   student s1(1, 24, "XYZ");                                 // Create the first student object
   s1.get_details();                                         // Display the details of the first student
   student s2(2, 25, "ABC");                                 // Create the second student object
   s2.get_details();                                         // Display the details of the second student

   return 0;
}
```

OUTPUT

"C:\Users\MSC\Desktop\Unversity\c c++\Program\c\w10 p3.exe"

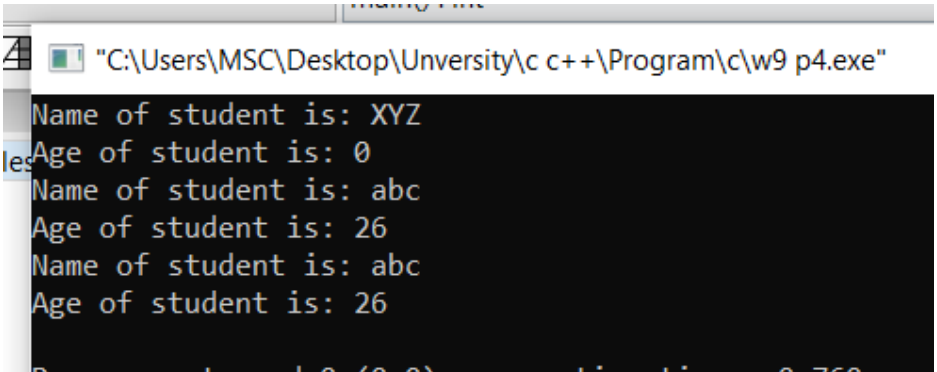
```
No. of objects created 1
Name: XYZ
Age: 1
Roll: 24
No. of objects created 2
Name: ABC
Age: 2
Roll: 25
No. of objects destroyed 2
No. of objects destroyed 1
```

Q42. Write a program in C++ demonstrates the use of function overloading

SOURCE CODE

```
#include <iostream>
using namespace std;

// Define a class to represent a sample
class sample {
public:
   // Function to sum two integers
   int sum(int m, int n);

   // Function to sum two doubles
   double sum(double m, double n);

   // Function to add 2 to an integer
   int sum(int m);
};

// Function to sum two integers
int sample::sum(int m, int n) {
   return m + n;
}

// Function to sum two doubles
double sample::sum(double m, double n) {
   return m + n;
}

// Function to add 2 to an integer
int sample::sum(int m) {
   return m + 2;
}

int main() {
   sample s1;  // Create an object of the sample class

   // Call the sum function with one integer argument
   cout << s1.sum(5) << endl;

   // Call the sum function with two double arguments
   cout << s1.sum(5.6, 7.8) << endl;

   // Call the sum function with two integer arguments
   cout << s1.sum(4, 5) << endl;

   return 0;
}
```

OUTPUT

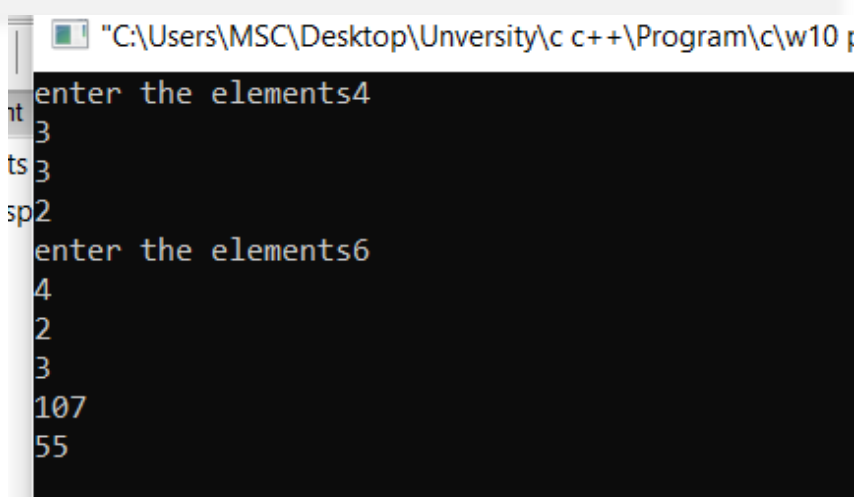


Q43. Write a C++ program to overload the '+' operator so that it can add two matrices.

SOURCE CODE

```
#include<iostream>                          // Include the input-output stream library
using namespace std;

class matrices {
   int a[2][2];                      // Matrix a
   int b[2][2];                      // Matrix b
   int c[2][2];                      // Matrix c

public:
   void get_elements();                     // Function to get elements of the matrix
   matrices operator +(matrices m2);    // Overload + operator to add matrices
   void display();                          // Function to display the matrix
};

void matrices::get_elements() {
   cout << "enter the elements";            // Prompt user to enter elements
   for(int i = 0; i < 2; i++) {
      for(int j = 0; j < 2; j++)
         cin >> a[i][j];                    // Input elements into matrix a
   }
}

void matrices::display() {
   for(int i = 0; i < 2; i++) {
      for(int j = 0; j < 2; j++)
         cout << a[i][j];             // Display elements of matrix a
      cout << endl;                   // New line after each row
   }
}

matrices matrices::operator+(matrices m2) {
   matrices m3;                     // Create a new matrix to store the result
   for(int i = 0; i < 2; i++) {
      for(int j = 0; j < 2; j++)
         m3.a[i][j] = a[i][j] + m2.a[i][j];     // Add corresponding elements
   }
   return m3;                        // Return the result matrix
}

int main() {
   matrices ob1, ob2;                       // Create two matrix objects
   ob1.get_elements();                       // Get elements for the first matrix
   ob2.get_elements();                      // Get elements for the second matrix
   ob1 = ob1 + ob2;                         // Add the two matrices
   ob1.display();                           // Display the result
}
```

OUTPUT



LAB EXERCISES

Week 11

Q44. Write a C++ program to overload the assignment operator
Q45. Write a C++ program to overload comparison operator == & operator !=
Q46. Write C++ program to overload the unary operator
Q47. Write a program in C++ which creates a single-inheritance hierarchy of Person, Employee and Teacher classes and creates instances of each class using new and stores them in an array of Person *

In this week

OPERATOR OVERLOADING in C++ allows you to redefine the way operators work for user-defined types (such as classes). This enables operators to be used with objects in a manner similar to built-in types.

INHERITANCE is a fundamental concept in object-oriented programming (OOP) that allows a class (derived class) to inherit properties and behaviors (methods) from another class (base class). This promotes code reusability and establishes a natural hierarchy between classes

Q44. Write a C++ program to overload the assignment operator.

SOURCE CODE

```
#include<iostream>
using namespace std;

class sample {                          // Class definition for sample
   int x, y;                            // Private data members
public:

   sample() {}                          // Default constructor

  sample(int i, int j) {            // Parameterized constructor
      x = i;
      y = j;
   }

                        // Function to display values of x and y
   void show() {
      cout << "x=  " << x << endl;
      cout << "y=  " << y << endl;
   }

      sample operator=(sample op2);       // Overloaded assignment operator
};

sample sample::operator=(sample op2) {    // Definition of overloaded assignment operator

   x = op2.x;
   y = op2.y;
   return *this;
}

int main() {
   sample ob1(10, 20), ob2(30, 40), ob3(50, 60);            // Create objects with initial values

   ob1.show();                      // Display values of ob1
   ob2.show();                      // Display values of ob2
   ob3.show();                       // Display values of ob3

   ob1 = ob2 = ob3;                  // Assign values of ob3 to ob2 and ob1

   ob1.show();                      // Display updated values of ob1
   ob2.show();                      // Display updated values of ob2

   return 0;
}
```

OUTPUT

```
x=  10
y=  20
x=  30
y=  40
x=  50
y=  60
x=  50
y=  60
x=  50
y=  60
```

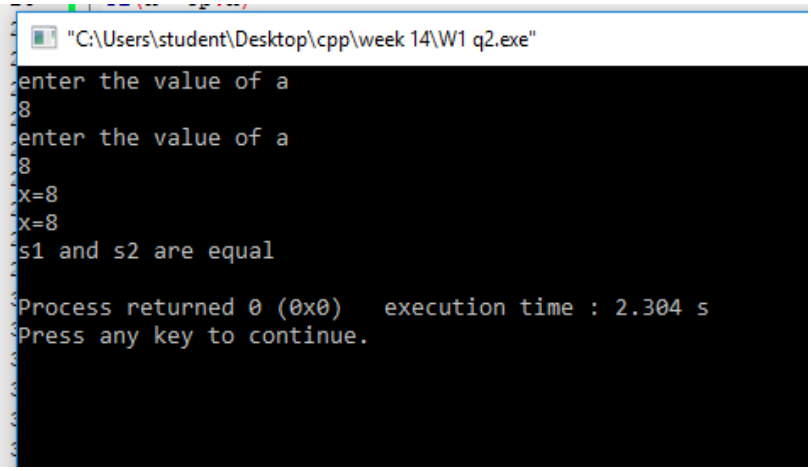
Q45. Write a C++ program to overload comparison operator == & operator !=

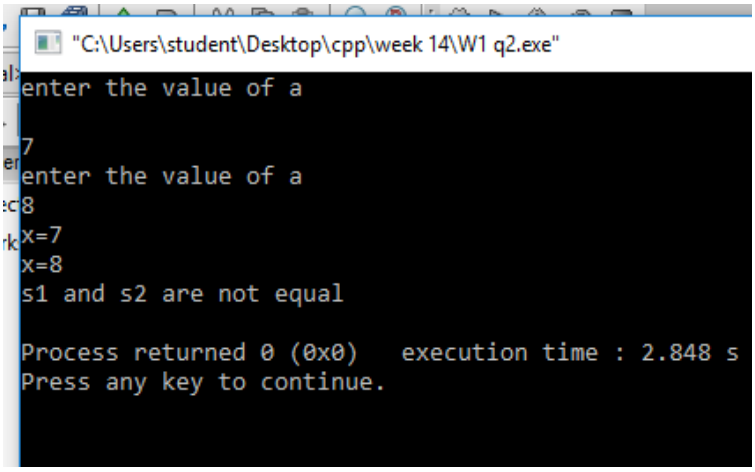
SOURCE CODE

overload comparison operator ==

```
#include<iostream>
using namespace std;

class sample {                    // Class definition for sample
   int x;                         // Private data member
public:

   void getdata() {   // Function to get data from user
      cout << "enter the value of a" << endl;
      cin >> x;
   }

   void show() {                                  // Function to display the value of x
      cout << "x=" << x << endl;
   }

   int operator==(sample op);                     // Overloaded equality operator
};

int sample::operator==(sample op) {     // Definition of overloaded equality operator

   int i;
   if (x == op.x)
      i = 1;                         // Return 1 if values are equal
   else
      i = 0;                                  // Return 0 if values are not equal
   return i;
}
```

OUTPUT



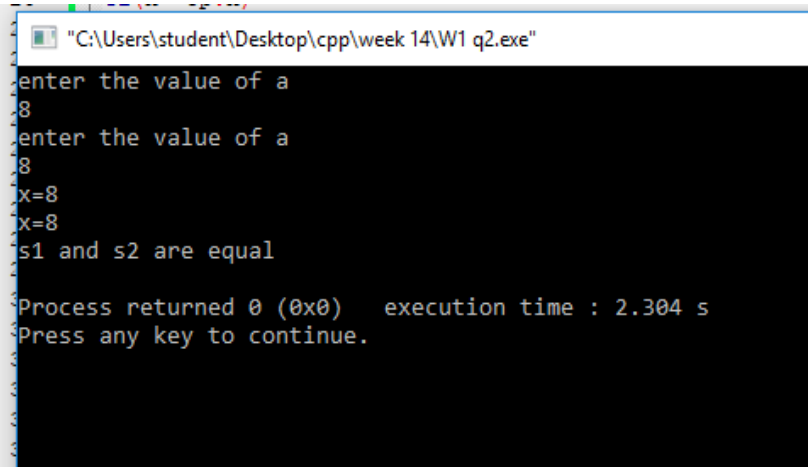


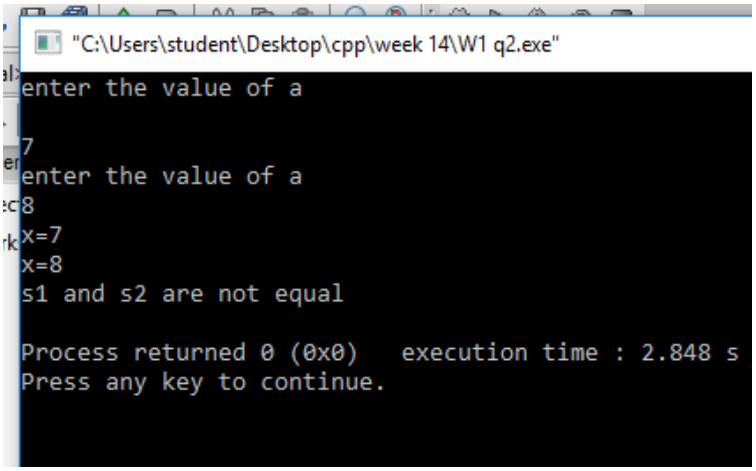
SOURCE CODE

overload comparison operator !=

```
#include<iostream>
using namespace std;

class sample {                    // Class definition for sample
   int x;                         // Private data member
public:

   void getdata() {   // Function to get data from user
      cout << "enter the value of a" << endl;
      cin >> x;
   }

   void show() {                                  // Function to display the value of x
      cout << "x=" << x << endl;
   }

   int operator==(sample op);                     // Overloaded equality operator
};

int sample::operator==(sample op) {     // Definition of overloaded equality operator

   int i;
   if (x == op.x)
      i = 1;                         // Return 1 if values are equal
   else
      i = 0;                                 // Return 0 if values are not equal
   return i;
}
```

OUTPUT



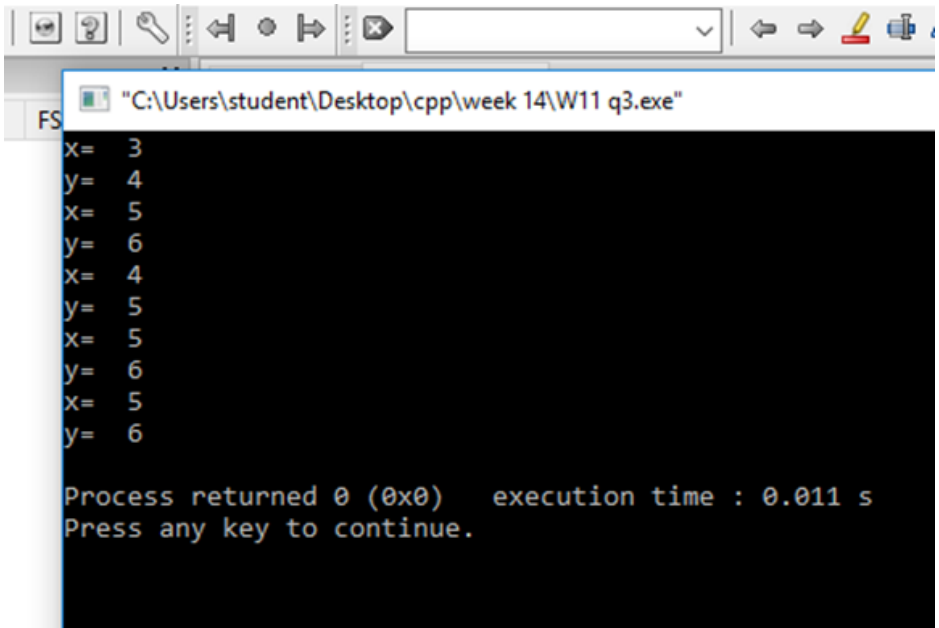


Q46. Write C++ program to overload the unary operator

SOURCE CODE

```
#include<iostream>
using namespace std;
                                        // Class definition for sample
class sample {
   int x, y;                                    // Private data members
public:
                                 // Default constructor
   sample() {}

                                 // Parameterized constructor
   sample(int i, int j) {
      x = i;
      y = j;
   }
                                // Function to display values of x and y
   void show() {
      cout << "x=  " << x << endl;
      cout << "y=  " << y << endl;
   }
                                // Overloaded prefix increment operator
   sample operator++();
};
                               // Definition of overloaded prefix increment operator
sample sample::operator++() {
   x++;                                                 // Increment x
   y++;                                                 // Increment y
   return *this;                                        // Return the updated object
}

int main() {
   sample ob1(3, 4), ob2(5, 6);                         // Create objects with initial values
   ob1.show();                                          // Display values of ob1
   ob2.show();                                          // Display values of ob2
   ++ob1;                                       // Prefix increment ob1
   ob1.show();                                          // Display updated values of ob1

   ob2 = ++ob1;                                 // Assign incremented ob1 to ob2
   ob1.show();                                          // Display updated values of ob1
   ob2.show();                                          // Display updated values of ob2

   return 0;
}
```

OUTPUT



Q47. Write a program in C++ which creates a single-inheritance hierarchy of Person, Employee and Teacher classes and creates instances of each class using new and stores them in an array of Person *

SOURCE CODE

```
#include <iostream>
#include <string>
using namespace std;
                                    // Base class Person
class Person {
protected:
   string name;                          // Name of the person
   int age;                              // Age of the person
public:
                                     // Constructor to initialize name and age
   Person(string n, int a) : name(n), age(a) {}
                                    // Virtual function to display person details
   virtual void display() {
      cout << "Name: " << name << ", Age: " << age << endl;
   }
};
                                    // Derived class Employee inheriting from Person
class Employee : public Person {
protected:
   float salary;                         // Salary of the employee
public:
                                      // Constructor to initialize name, age, and salary
   Employee(string n, int a, float s) : Person(n, a), salary(s) {}

                                     // Overridden function to display employee details
   void display() override {
      cout << "Name: " << name << ", Age: " << age << ", Salary: " << salary << endl;
   }
};
                                    // Derived class Teacher inheriting from Employee
class Teacher : public Employee {
private:
   string subject;                       // Subject taught by the teacher
public:
                                    // Constructor to initialize name, age, salary, and subject
   Teacher(string n, int a, float s, string sub) : Employee(n, a, s), subject(sub) {}

                                     // Overridden function to display teacher details
   void display() override {
      cout << "Name: " << name << ", Age: " << age << ", Salary: " << salary << ", Subject: " << subject << endl;
   }
};
int main() {
                                     // Array of Person* to store instances
   Person* people[3];
                                      // Creating instances using new and storing them in the array
   people[0] = new Person("shariq", 30);
   people[1] = new Employee("mohizeen", 40, 50000);
   people[2] = new Teacher("Manzoor sir", 35, 60000, "CPP");
                                     // Displaying information of each person
   for (int i = 0; i < 3; i++) {
      people[i]->display();
   }
// Deleting dynamically allocated memory
   for (int i = 0; i < 3; i++) {
      delete people[i];
   }
   return 0;
```

OUTPUT

```
"C:\Users\student\Desktop\cpp\week 14\W11 q4.exe"
Name: shariq, Age: 30
Name: mohizeen, Age: 40, Salary: 50000
Name: Manzoor sir, Age: 35, Salary: 60000, Subject: CPP

Process returned 0 (0x0)   execution time : 0.023 s
Press any key to continue.
```

LAB EXERCISES

Week 12

Q48. Write a program in C++ which creates a multiple inheritance hierarchy of Teacher classes derived from bot Person, Employee classes. Each class must implement a Show() member function and utilize scope resolution operator

Q49. Write a C++ program that demonstrates the concept of function overriding

Q50. Write a C++ program to show inheritance using different levels.

Q51. Write a C++ program to demonstrate the concepts of abstract class and inner class

In this week

METHOD OVERRIDING in C++ occurs when a derived class provides a specific implementation of a function that is already defined in its base class. The function in the derived class must have the same name, return type, and parameters as the one in the base class.

Q48. Write a program in C++ which creates a multiple inheritance hierarchy of Teacher classes derived from bot Person, Employee classes. Each class must implement a Show() member function and utilize scope resolution operator

SOURCE CODE

```
#include <iostream>
using namespace std;
#include <string.h>

class person {                          // Base class person
protected:
   int age;                             // Protected data member for age
   char name[50]                ; // Protected data member for name
public:

   person(int a, char *n) {      // Constructor to initialize age and name
      age = a;
      strcpy(name, n);
   }
      void show() {                                  // Function to display person details
      cout << "name: " << name << endl;
      cout << "age:  " << age << endl;
   }
};

class Employee {                // Base class Employee
protected:
   float salary;                        // Protected data member for salary
public:
   Employee(int s) {                // Constructor to initialize salary
      salary = s;
   }
   void show() {                    // Function to display employee details
      cout << "salary: " << salary << endl;
   }
};

// Derived class Teacher inheriting from person and Employee
class Teacher : public person, public Employee {
protected:
   char area[50];                   // Protected data member for research area
public:
// Constructor to initialize all data members
   Teacher(int a, char *n, int s, char *ar) : Employee(s), person(a, n) {
      strcpy(area, ar);
   }
   void show() {                    // Function to display teacher details
      person::show();                                // Call show() of person class
      Employee::show();                              // Call show() of Employee class
      cout << "research_area: " << area << endl;
   }
};

int main() {
      Teacher T1(21, "ABC", 7879, "Comp");        // Create a Teacher object and display its details
   T1.show();
   return 0;
}
```

OUTPUT

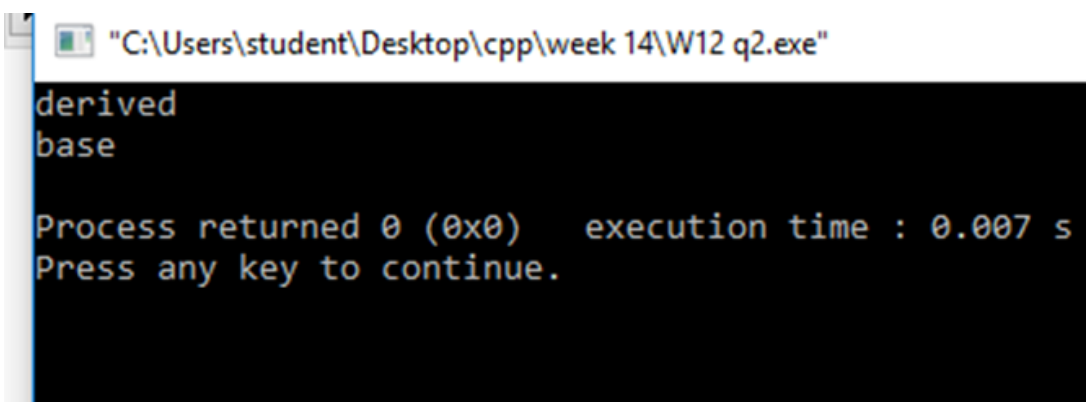
```
"C:\Users\student\Desktop\cpp\week 14\W12 q1.exe"
name: mOZIN
age:  21
salary: 7879
research_area: Comp

Process returned 0 (0x0)   execution time : 0.011
Press any key to continue.
```

Q49. Write a C++ program that demonstrates the concept of function overriding

SOURCE CODE

```
#include<iostream>
using namespace std;
class Base {                                // Base class
public:
      void show() {                         // Function to show base class message
      cout << "base " << endl;
   }
};
class Derived : public Base {           // Derived class inheriting from Base
public:
   void show() {                    // Function to show derived class message
      cout << "derived" << endl;
   }
};
int main() {
   Derived d;                     // Create an object of Derived class
   d.show();                     // Invokes show() in derived class
   d.Base::show();            // Invokes show() in base class using scope resolution operator
   return 0;
}
```

OUTPUT



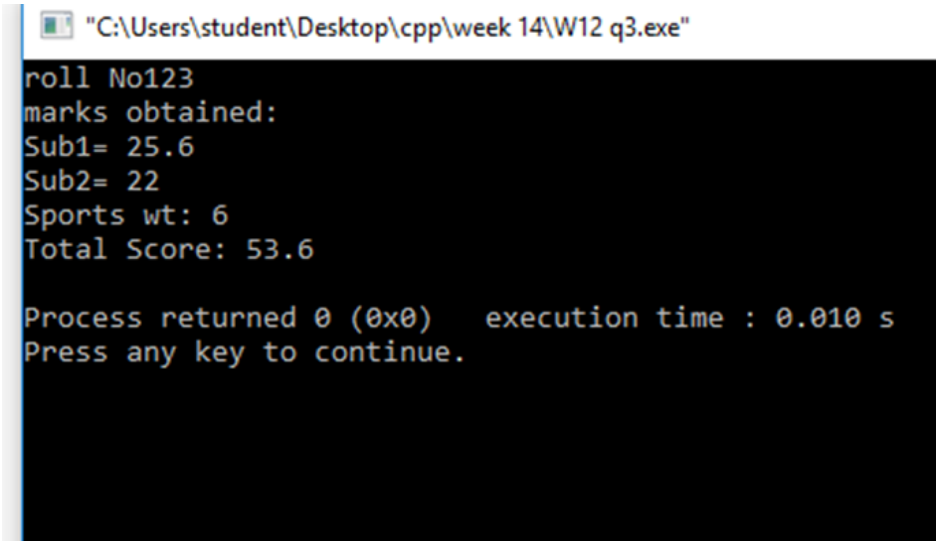
Q50. Write a C++ program to show inheritance using different levels

SOURCE CODE

```
#include<iostream>
using namespace std;

class student {                    // Base class for student details
protected:
   int r_number;                   // Roll number
public:
      void get_number(int a) {                    // Function to set roll number
      r_number = a;
   }
   void put_number() {                              // Function to display roll number
      cout << "roll No: " << r_number << endl;
   }
};

class test : public student {                      // Derived class for test marks
protected:
   float sub1, sub2;                                // Marks for two subjects
public:
   void get_marks(float x, float y) {                // Function to set marks
      sub1 = x;
      sub2 = y;
   }
   void put_marks() {                        // Function to display marks
      cout << "marks obtained: " << endl;
      cout << "Sub1 = " << sub1 << endl;
      cout << "Sub2 = " << sub2 << endl;
   }
};

class sports {                     // Class for sports score
protected:
   float score;                    // Sports score
public:
   void get_score(float s) {                 // Function to set sports score
      score = s;
   }
   void put_score() {                        // Function to display sports score
      cout << "Sports wt: " << score << endl;
   }
};

// Derived class for result, inheriting from test and sports
class result : public test, public sports {
   float total; // Total score
public:

   void display();                // Function to display total result
};

// Implementation of display function
void result::display() {
   total = sub1 + sub2 + score;                          // Calculate total score
   put_number();                          // Display roll number
   put_marks();                            // Display marks
   put_score();                           // Display sports score
   cout << "Total Score: " << total << endl;              // Display total score
}
```

```
int main() {
   result student_1;                          // Create result object
   student_1.get_number(123);          // Set roll number
   student_1.get_marks(25.6, 22.0);           // Set marks
   student_1.get_score(6.0);                  // Set sports score
   student_1.display();                       // Display result
   return 0;
}
```

OUTPUT

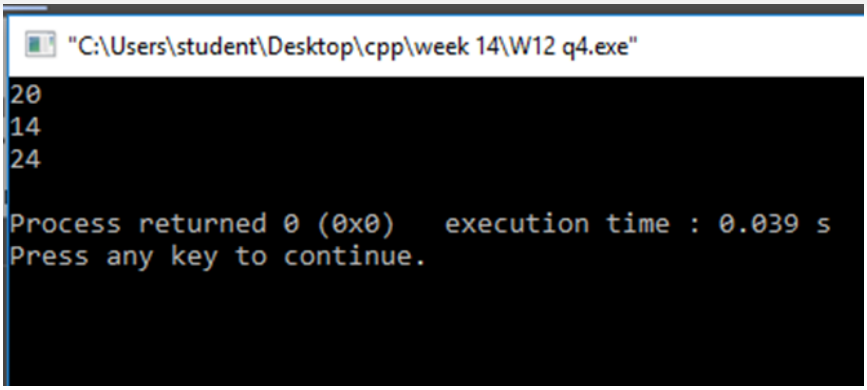


Q51. Write a C++ program to demonstrate the concepts of abstract class and inner class.

SOURCE CODE

```
#include <iostream>
using namespace std;
class number {                          // Abstract base class
protected:
   int val;                                     // Protected data member
public:
   void setval(int i) {
      val = i;                                  // Set the value of val
   }
   virtual void show() = 0;                     // Pure virtual function
};
class hextype : public number {         // Derived class for hexadecimal representation
public:
   void show() override {
      cout << hex << val << endl;       // Display value in hexadecimal
   }
};
class dectype : public number {     // Derived class for decimal representation
public:
   void show() override {
      cout << val << endl;          // Display value in decimal
   }
};
class octtype : public number {     // Derived class for octal representation
public:
   void show() override {
      cout << oct << val << endl;       // Display value in octal
   }
};
int main() {
   dectype d;                       // Decimal type object
   hextype h;                        // Hexadecimal type object
   octtype o;                       // Octal type object

   d.setval(20);                    // Set value for decimal object
   d.show();                         // Show value in decimal

   h.setval(20);                    // Set value for hexadecimal object
   h.show();                        // Show value in hexadecimal

   o.setval(20);                     // Set value for octal object
   o.show();                         // Show value in octal

   return 0;
}
```

OUTPUT



LAB EXERCISES

Week 13

Q52. Write a C++ program to demonstrate the use of virtual functions and polymorphism
Q53. Write a C++ program to demonstrate the use of pure virtual functions and virtual destructors.
Q54. Write a C++ program to swap data using function templates
Q55. Write a C++ program to create a simple calculator which can add, subtract, multiply and divide two numbers using class template

In this week

VIRTUAL DESTRUCTORS: ensure that the destructor of the derived class is called when an object is deleted through a base class pointer. This prevents resource leaks and undefined behavior.

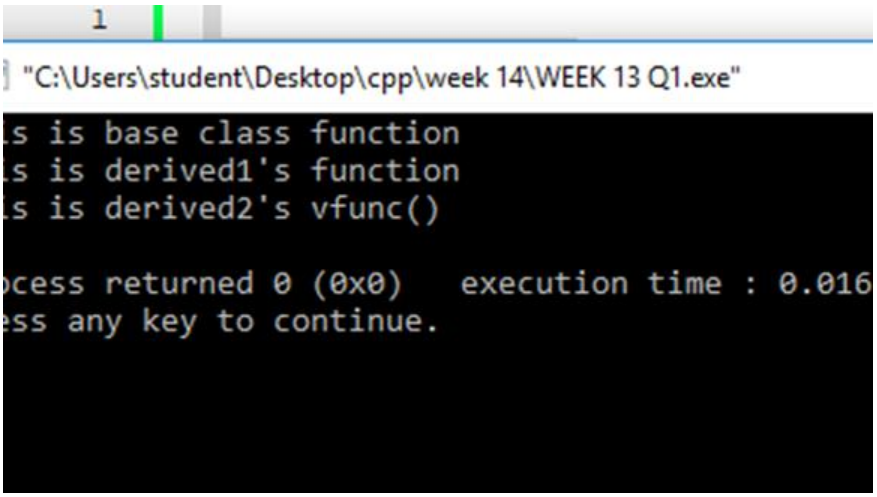
VIRTUAL DESTRUCTORS: When delete obj is called, the Derived class's destructor is invoked first, followed by the Base class's destructor, ensuring proper cleanup

Q52. Write a C++ program to demonstrate the use of virtual functions and polymorphism

SOURCE CODE

```
#include<iostream>
using namespace std;

class base {                                    // Base class with a virtual function
public:
   virtual void vfunc() {
      cout << "This is base class function" << endl;
   }
};
class derived1 : public base {                  // Derived class 1 overriding the virtual function
public:
   void vfunc() override {
      cout << "This is derived1's function" << endl;
   }
};
class derived2 : public base {                          // Derived class 2 overriding the virtual function
public:
   void vfunc() override {
      cout << "This is derived2's vfunc()" << endl;
   }
};
int main() {
   base *p, b;                          // Pointer to base class and base class object
   derived1 d1;                         // Derived class 1 object
   derived2 d2;                        // Derived class 2 object

   p = &b;                             // Point to Base
   p->vfunc();                          // Calls base class function


   p = &d1;                             // Point to derived1
   p->vfunc();                          // Calls derived1's function

   p = &d2;                            // Point to derived2
   p->vfunc();                         // Calls derived2's function

   return 0;
}
```

OUTPUT



Q53. Write a C++ program to demonstrate the use of pure virtual functions and virtual destructors.

SOURCE CODE

```
#include<iostream>
using namespace std;
class number {                                  // Base class with a pure virtual function
protected:
   int val;                                     // Protected data member to store value
public:
   void setval(int i) {
      val = i; // Set the value of val
   }
   virtual void show() = 0;                     // Pure virtual function
   virtual ~number() {
      cout << "number object deleted" << endl;          // Destructor message
   }
};
class hextype : public number {       // Derived class for hexadecimal representation
public:
   void show() override {
      cout << hex << val << endl;                  // Display value in hexadecimal
   }
   ~hextype() {
      cout << "hextype object deleted" << endl;                    // Destructor message
   }
};
class dectype : public number {       // Derived class for decimal representation
public:
   void show() override {
      cout << val << endl;                 // Display value in decimal
   }
   ~dectype() {
      cout << "dectype object deleted" << endl;          // Destructor message
   }
};
class octtype : public number {    // Derived class for octal representation
public:
   void show() override {
      cout << oct << val << endl;                          // Display value in octal
   }
   ~octtype() {
      cout << "octtype object deleted" << endl;                                        // Destructor message
   }
};
int main() {
   number *ptr;                         // Pointer to base class
   dectype d;                           // Decimal type object
   hextype h;                           // Hexadecimal type object
   octtype o;                           // Octal type object
   ptr = &d;                      // Point to dectype object
   ptr->setval(20);                     // Set value
   ptr->show();                         // Show value in decimal

   ptr = &h;                            // Point to hextype object
   ptr->setval(20);                     // Set value
   ptr->show();                         // Show value in hexadecimal
   ptr = &o;                      // Point to octtype object
   ptr->setval(20);                     // Set value
   ptr->show();                         // Show value in octal
   return 0;
}
```

OUTPUT

```
"C:\Users\student\Desktop\cpp\week 14\WEEK 13 Q1.exe"
20
14
24
octtype object deleted
number object deleted
hextype object deleted
number object deleted
dectype object deleted
number object deleted

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Q54. Write a C++ program to swap data using function templates.

SOURCE CODE

```
#include<iostream>
using namespace std;

template <class T>                    // Template function to swap two arguments
void swapargs(T& x, T& y) {
   T temp;
   temp = x;                          // Store the value of x in temp
   x = y;                             // Assign the value of y to x
   y = temp;                          // Assign the value of temp (original x) to y
}

void fun(int m, int n) {              // Function to demonstrate the swap
   cout << "m and n before swap: " << m << " " << n << endl;
   swapargs(m, n);                    // Call the template function to swap m and n
   cout << "m and n after swap: " << m << " " << n << endl;
}

int main() {
   int i = 10, j = 20;
   fun(i, j);                         // Call the function to demonstrate swapping
   return 0;

}
```

OUTPUT

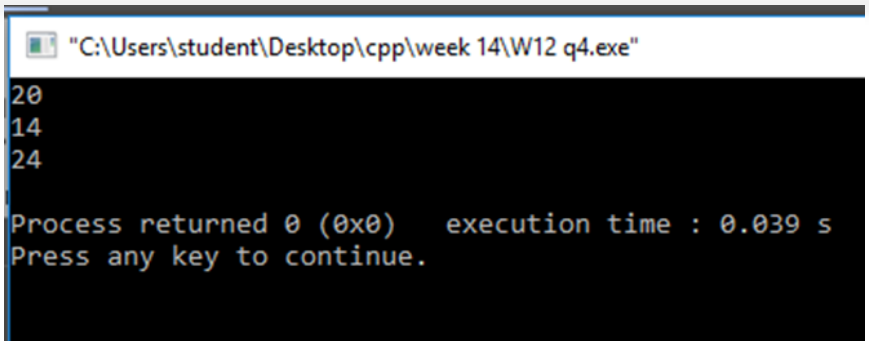
```
"C:\Users\student\Desktop\cpp\week 14\W13 Q3.exe"
m and n before swap: 10 20
m and n after swap: 20 10

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Q55. Write a C++ program to create a simple calculator which can add, subtract, multiply and divide two numbers using class template.

SOURCE CODE

```
#include <iostream>
using namespace std;
template <class T>                    // Template class for Calculator
class Calculator {
private:
   T num1, num2;                              // Private data members to store the numbers
public:
      Calculator(T n1, T n2) : num1(n1), num2(n2) {}  // Constructor to initialize the numbers
   T add() {                                        // Function to add the numbers
      return num1 + num2;
   }
   T subtract() {                          // Function to subtract the numbers
      return num1 - num2;
   }
      T multiply() {                          // Function to multiply the numbers
      return num1 * num2;
   }
      T divide() {                            // Function to divide the numbers
      if (num2 != 0)
         return num1 / num2;
      else {
         cout << "Error: Division by zero!" << endl;
         return 0;
      }   } };
int main() {
                        // Create Calculator objects for different data types
   Calculator<int> intCalc(10, 5);                      // Integer calculator
   Calculator<float> floatCalc(10.5, 2.5); // Float calculator
              // Perform operations using integer calculator
   cout << "Integer Operations:" << endl;
   cout << "10 + 5 = " << intCalc.add() << endl;
   cout << "10 - 5 = " << intCalc.subtract() << endl;
   cout << "10 * 5 = " << intCalc.multiply() << endl;
   cout << "10 / 5 = " << intCalc.divide() << endl;
                        // Perform operations using float calculator
   cout << "Float Operations:" << endl;
   cout << "10.5 + 2.5 = " << floatCalc.add() << endl;
   cout << "10.5 - 2.5 = " << floatCalc.subtract() << endl;
   cout << "10.5 * 2.5 = " << floatCalc.multiply() << endl;
   cout << "10.5 / 2.5 = " << floatCalc.divide() << endl;
   return 0;
}
```

OUTPUT



LAB EXERCISES

Week 14

Q56. Write a C++ program to demonstrate the concept of exception handling.

Q57. Write a C++ program to create a custom exception. Define a class with appropriate data members and member functions which opens an input and output file, checks each one for being open, and then reads name, age, salary of a person from the input file and stores the information in an object, increases the salary by a bonus of 10% and then writes the person object to the output file. It continues until the input stream is no longer good

In this week

EXCEPTION HANDLING

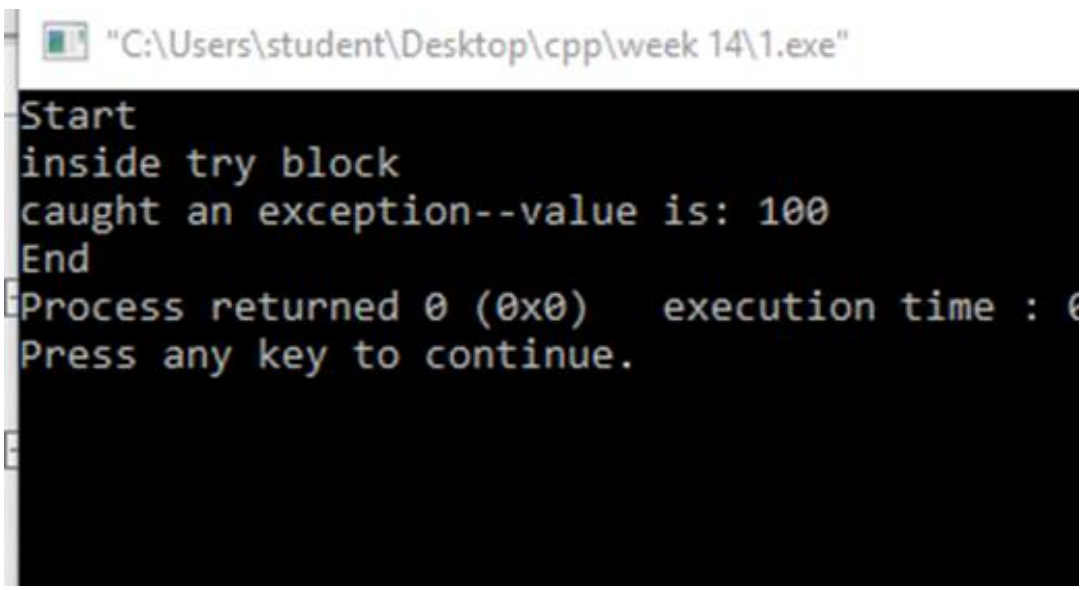
is a mechanism to handle runtime errors, ensuring the program can manage unexpected situations gracefully without crashing. It uses three main keywords: try, catch, and throw.

Q56. Write a C++ program to demonstrate the concept of exception handling.

SOURCE CODE

```
#include<iostream>
using namespace std;
int main() {
   cout << "Start\n";                          // Print start message
   try { // Start a try block
      cout << "inside try block\n";             // Print inside try block message
      throw 100; // Throw an error
      cout << "This will not execute";          // This line will not be executed
   } catch (int i) {                            // Catch the error
      cout << "caught an exception--value is: "; // Print caught exception message
      cout << i << "\n";                        // Print the exception value
   }
   cout << "End";                              // Print end message
   return 0;
}
```

OUTPUT



Q53. Write a C++ program to create a custom exception. Define a class with appropriate data members and member functions which opens an input and output file, checks each one for being open, and then reads name, age, salary of a person from the input file and stores the information in an object, increases the salary by a bonus of 10% and then writes the person object to the output file. It continues until the input stream is no longer good

SOURCE CODE

```
#include <iostream>
#include <fstream>
#include <string>
#include <exception>

using namespace std;

// Custom exception class
class FileException : public exception {
private:
   string message;
public:
   FileException(const string& msg) : message(msg) {}
   const char* what() const noexcept override {
      return message.c_str();
   }
};

// Person class to store person details
class Person {
private:
   string name;
   int age;
   float salary;
public:
   Person(const string& name, int age, float salary) : name(name), age(age), salary(salary) {}
   void increaseSalary(float bonus) {
      salary += salary * bonus / 100;
   }
   void writeToFile(ofstream& outFile) {
      outFile << name << " " << age << " " << salary << endl;
   }
   friend ifstream& operator>>(ifstream& inFile, Person& person);
};

// Overload >> operator to read Person object from file
ifstream& operator>>(ifstream& inFile, Person& person) {
   inFile >> person.name >> person.age >> person.salary;
   return inFile;
}

int main() {
   ifstream inFile("input.txt");
   ofstream outFile("output.txt");

   // Check if files are open
   if (!inFile.is_open()) {
      throw FileException("Error opening input file");
   }
```

```
    if (!outFile.is_open()) {
        throw FileException("Error opening output file");
    }

    Person person("", 0, 0.0);
    while (inFile >> person) {
        person.increaseSalary(10);          // Increase salary by 10%
        person.writeToFile(outFile);                    // Write updated person to output file
    }

    inFile.close();
    outFile.close();

    cout << "Processing completed successfully." << endl;
    return 0;
}
```

OUTPUT