1.1

Write a program to demonstrate the use of Output statements that draws any object of your choice e.g. Christmas Tree using '*'

```
#include <stdio.h>

int main(){

    printf("     *\n");
    printf("    ***\n");
    printf("   *****\n")
    printf("     *\n");
    printf("    ***\n");
    printf("   *****\n");
    printf("     *\n");
    printf("     *\n");
    printf("     *\n");

    return 0;
}
```

```
     *
    ***
   *****
     *
    ***
   *****
     *
     *
     *
```

1.2

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

```
#include <stdio.h>

int main()
{

    int monthNumber = 0;
    printf("Enter month number: ");
    scanf("%d", &monthNumber);

    switch (monthNumber)
    {
    case 1:
        printf("January\n");
        break;
    case 2:
        printf("February\n");
        break;
    case 3:
        printf("March\n");
        break;
    case 4:
        printf("April\n");
        break;
    case 5:
        printf("May\n");
        break;
    case 6:
        printf("June\n");
        break;
    case 7:
        printf("July\n");
        break;
    case 8:
        printf("August\n");
        break;

    case 9:
        printf("September\n");
        break;
    case 10:
        printf("October\n");
        break;
    case 11:
        printf("November\n");
        break;
    case 12:
        printf("December\n");
        break;
    default:
        printf("Not a valid month number\n");
        break;
    }
    return 0;
}
```

Another approach

```
int main()
{

    int monthNumber = 0;
    printf("Enter month number: ");
    scanf("%d", &monthNumber);

    char *monthNames[12] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};

    if(monthNumber >= 1 && monthNumber <= 12){
        printf("%s", monthNames[monthNumber - 1]);
    }else {
        printf("Not a valid Number");
    }
    return 0;
}
```

```
Enter month number: 5
May
```

```
Enter month number: -10
Not a valid month
```

1.3

Write a program that demonstrates the use of various input statements like getchar(), getch(), scanf().

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

int main(){

    char ch = 'A';
    printf("initial value of ch: %c\n", ch);
    printf("Enter char value via getchar(): ");
    ch = getchar();
    printf("You entered: %c\n", ch);

    fflush(stdin);

    printf("Enter char value vai scanf(): ");
    scanf("%c", &ch);
    printf("You entered: %c\n", ch);


    printf("Enter char via getch()\n");
    ch = getch();
    printf("You entered: %c\n", ch);


    return 0;
}
```

```
initial value of ch: A
Enter char value via getchar(): a
You entered: a
Enter char value vai scanf(): b
You entered: b
Enter char via getch()
You entered: c
```

1.4

Write a program to demonstrate the overflow and underflow of various datatype and their resolution?

```
#include <stdio.h>
#include <limits.h>
#include <float.h>

int main(){

    printf("Data Type bytes range Max+1 Min-1\n");
    printf("int %d %d to %d  %d  %d\n", sizeof(int), INT_MIN, INT_MAX, INT_MAX + 1,
INT_MIN - 1);

    printf("unsigned int %d %d to %lu   %d   %lu\n", sizeof(unsigned int), 0, UINT_MAX,
UINT_MAX + 1, (unsigned int)(-1));

    printf("long long int %d %lld to %lld %lld  %lld\n", sizeof(long long), LLONG_MIN,
LLONG_MAX, LLONG_MAX + 1, LLONG_MIN - 1);

    printf("char %d %d to %d   %d   %d\n", sizeof(char), CHAR_MIN, CHAR_MAX,
(char)(CHAR_MAX + 1), (char)(CHAR_MIN - 1));

    printf("u char %d %d to %d   %d   %d\n", sizeof(unsigned char), 0, UCHAR_MAX,
(unsigned char)(UCHAR_MAX + 1), (unsigned char)-1);

    return 0;
}
```

```
Data Type   bytes       range                          Max+1             Min-1
Int         4           -2147483648 to 2147483647      -2147483648 2147483647
unsigned int    4       0 to 4294967295                0                 4294967295
long long int   8     -9223372036854775808 to 9223372036854775807
-9223372036854775808                    9223372036854775807
Char        1           -128 to 127                    -128              127
u char     1          0 to 255                       0                255
```

2.1

Write a program to demonstrate the precedence of various operators.

```
#include <stdio.h>

int main(){

    printf("7 + 3 * 2 = \t\t%d\n", 7 + 3 * 2);
    printf("15 / 5 * 2 = \t\t%d\n",15 / 5 * 2);
    printf("(15 / 5) * 2 = \t\t%d\n",(15 / 5) * 2);
    printf("20+10*15/5 = \t\t%d\n", 20+10*15/5);
    printf("(20+10)*15/5 = \t\t%d\n", (20+10)*15/5);
    printf("(20+10)*(15/5) = \t%d\n", (20+10)*(15/5));
    int x = 1, y = 100;
    printf("x++ = \t\t\t%d\n", x++);
    printf("x++ + ++x = \t\t%d\n", x++ + ++x);
    printf("x-- + ++x * x-- = \t%d\n", x-- + ++x * x--);
    printf("x > 50 && y > 50 = \t%d\n",x > 50 && y > 50);

    return 0;
}
```

```
7 + 3 * 2 =             13
15 / 5 * 2 =            6
(15 / 5) * 2 =          6
20+10*15/5 =            50
(20+10)*15/5 =          90
(20+10)*(15/5) =        90
x++ =                   1
x++ + ++x =             6
x-- + ++x * x-- =       16
x > 50 && y > 50 =      0
```

2.2

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

```
#include <stdio.h>

int main(){

    printf("Ascending Order\n");
    for (int i = 0; i < 20; i++){
        printf("%d, ", i);
    }
    printf("\nDescending Order\n");
    for (int i = 20; i>=0; i--){
        printf("%d, ", i);
    }
        return 0;
}
```

```
Ascending Order
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
Descending Order
20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0,
```

2.3

Write a program to generate pascal’s triangle.

```
#include <stdio.h>

void printPascal(int n)
{
    for (int line = 1; line <= n; line++) {
        for (int space = 1; space <= n - line; space++)
            printf("  ");
        int coef = 1;
        for (int i = 1; i <= line; i++) {
            printf("%4d", coef);
            coef = coef * (line - i) / i;
        }
        printf("\n");
    }
}

int main()
{
    int n = 5;
    printf("Enter the size of pascal\'s triangle: ");
    scanf("%d", &n);
    printPascal(n);
    return 0;
}
```

```
Enter the size of pascal's triangle: 5
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

2.4

Write a program to reverse the digits of a given number. For example, the number 9876 should be returned as 6789.

```
#include <stdio.h>

int reverse(int num){
    int reversedNum = 0;
    while(num != 0){
        reversedNum = (reversedNum * 10) + (num % 10);
        num /= 10;
    }
    return reversedNum;
}

int main(){

    int num = 0;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Reversed Number is: %d", reverse(num));
    return 0;
}
```

```
Enter a number: 5406
Reversed Number is: 6045
```

3.1

Write a program to convert an amount (upto billion 10e9) in figures to equivalent amount in words.

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

char *threeFigureWord(int figure, char *words);
char *toWords(unsigned long long figure, char
*words);


int main()
{
    char words[200] = "";
    unsigned long long int figure = 0;
    printf("Enter a number : ");
    scanf("%lld", &figure);
    printf("%s", toWords(figure, words));
    return 0;
}

char *toWords(unsigned long long figure, char
*words)
{
    int billions = figure / 1000000000;
    figure %= 1000000000;
    if (billions)
    {
        threeFigureWord(billions, words);
        strcat(words, " billion");
    }

    int millions = figure / 1000000;
    figure %= 1000000;
    if (millions)
    {
        strcat(words, " ");
        threeFigureWord(millions, words);
        strcat(words, " million");
    }

    int thousands = figure / 1000;
    figure %= 1000;
    if (thousands)
    {
        strcat(words, " ");
        threeFigureWord(thousands, words);
        strcat(words, " thousand");
    }

    if (figure)
    {
        strcat(words, " ");
        threeFigureWord(figure, words);
    }
    return words;
}


char *threeFigureWord(int figure, char *words)
{
    char *units[9] = {"one", "two", "three",
"four", "five", "six", "seven", "eight", "nine"};

    char *tens[9] = {"ten", "twenty", "thirty",
"forty", "fifty", "sixty", "seventy", "eighty",
"ninety"};

    char *teens[10] = {"ten", "eleven", "twelve",
"thirteen", "fourteen", "fifteen", "sixteen",
"seventeen", "eighteen", "nineteen"};

    if (figure / 100)
    {
        strcat(words, units[(figure / 100) - 1]);
        strcat(words, " hundred");
        figure %= 100;
    }

    if (figure / 10)
    {
        strcat(words, " ");
        if (figure / 10 == 1)
        {
            strcat(words, teens[figure % 10]);
            figure = 0;
        }
        else
        {
            strcat(words, tens[(figure / 10) -
1]);
            figure %= 10;
        }
    }

    if (figure)
    {
        strcat(words, " ");
        strcat(words, units[(figure % 10) - 1]);
    }
    return words;
}
```

```
Enter a number : 489543234312
four hundred eighty nine billion five hundred
forty three million two hundred thirty four
thousand three hundred twelve
```

3.2

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

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0

int isPrime(int num)
{

    for (int i= 2; i * i <= num; i++)
    {
        if (num % i == 0)
        {
            return FALSE;
        }
    }
    return TRUE;
}
int main()
{

    int sum = 0;
    for (int i = 101; i <= 500; i += 2)
    {
        if (isPrime(i))
        {
            sum += i;
        }
    }
    printf("sum of all primes between 100 and 500 is : %d\n", sum);
    return 0;
}
```

```
sum of all primes between 100 and 500 is : 20476
```

3.3

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

```
#include <stdio.h>

int main(){

    char str[100] = "";
    printf("Enter a string: ");
    gets(str);
    printf("You entered: %s\n", str);
    return 0;
}
```

```
Enter a string: Hello World
You entered: Hello World
```

3.4

Write a program to input 20 arbitrary numbers in a one-dimensional array. Calculate Frequency of each number. Print the number and its frequency in a tabular form.

```
#include <stdio.h>
#define SIZE 20

int main()
{

    int nums[SIZE];
    printf("Enter %d numbers: \n",
SIZE);
    for (int i = 0; i < SIZE; i++)
    {
        scanf("%d", &nums[i]);
    }

    int freq[SIZE][2];
    int index = 0;

    for (int i = 0; i < SIZE; i++)
    {
        int flag = 1;
        for (int j = 0; j < index; j++)
        {
            if (nums[i] == freq[j][0])
            {
                flag = 0;
                freq[j][1] = freq[j][1]
+ 1;
            }
        }
        if (flag)
        {
            freq[index][0] = nums[i];
            freq[index++][1] = 1;
        }
    }

    printf("Number\t|   Frequency\n");

    printf("____________________\n");

    for (int i = 0; i < index; i++)
    {
        printf("%d\t|   %d\n",
freq[i][0], freq[i][1]);
    }

    return 0;
}
```

```
Enter 20 numbers:
1 2 2 2 2 2 3 3 3 3 4 4 3 2 1 3 4 5 6 7
Number  |   Frequency
____________________
1       |   2
2       |   6
3       |   6
4       |   3
5       |   1
6       |   1
7       |   1
```

4.1

Write a C function to remove duplicates from an ordered array. For example, if input array contains 10,10,10,30,40,40,50,80,80,100 then output should be 10,30,40,50,80,100.

```
#include <stdio.h>

/// Returns new array length
int removeDuplicates(int nums[], int n)
{
    int index = 0;
    for (int i = 1; i < n; i++)
    {
        if (nums[i] != nums[index])
        {
            nums[++index] = nums[i];
        }
    }
    return index;
}

int main()
{
    int n;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    int nums[n];
    printf("Enter elements of array in
order\n", n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &nums[i]);
    }

    int newLength =
removeDuplicates(nums, n);

    for (int i = 0; i <= newLength; i++)
    {
        printf("%d, ", nums[i]);
    }

    return 0;
}
```

```
Enter the size of array: 10
Enter elements of array in order
10 10 10 30 40 40 50 80 80 100
10, 30, 40, 50, 80, 100,
```

4.2

Write a program which will arrange the positive and negative numbers in a one-dimensional array in such a way that all positive numbers should come first and then all the negative numbers will come without changing original sequence of the numbers. Example: Original array contains: 10,-15,1,3,-2,0,-2,-3,2,-9 Modified array: 10,1,3,0,2,-15,-2,-2,-3,-9

```
#include <stdio.h>

int main(){

    int n = 0;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    int nums[n];
    printf("Enter elements of array\n",
n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &nums[i]);
    }

    /*Save positives in another array */
    int positives[n];
    int pIndex = 0;
    for (int i = 0; i < n; i++){
        if (nums[i] >= 0){
            positives[pIndex++] =
nums[i];
        }
    }

    /* relocate negative numbers to the
end of the array */

    int nIndex = n - 1;
    for (int i = n - 1; i >= 0; i--){
        if(nums[i] < 0){
            nums[nIndex--] = nums[i];
        }
    }

    /* add all positive numbers back to
the array from auxilary positives array
*/
    for (int i = 0; i < pIndex; i++){
        nums[i] = positives[i];
    }

    for (int i = 0; i < n; i++){
        printf("%d, ", nums[i]);
    }

    return 0;
}
```

```
Enter the size of array: 10
Enter elements of array
10 -15 1 3 -2 0 -2 -3 2 -9
10, 1, 3, 0, 2, -15, -2, -2, -3, -9,
```

4.3

Write a program to compute addition, multiplication, and transpose of a 2-D array.

```
#include <stdio.h>
#include <stdlib.h>
#define ROWS 3
#define COLS 3

void printMatrix(int matrix[ROWS][COLS],
char* message){
    printf("\n*** %s ***\n", message);
    for (int i = 0; i < ROWS; i++){
        for (int j = 0; j < COLS; j++){
            printf("%d, ",
matrix[i][j]);
        }
        printf("\n");
    }
}



int main(){

    int matrix1[ROWS][COLS] = {
                            {1, 2, 3},
                            {5, 6, 7},
                            {9, 4, 2}};
    int matrix2[ROWS][COLS] = {
                            {2, 2, 5},
                            {5, 4, 1},
                            {1, 1, 7}};

    int addition[ROWS][COLS];
    int transpose[COLS][ROWS];
    int multiplication[ROWS][COLS];
    for (int i = 0; i < ROWS; i++){
        for (int j = 0; j < COLS; j++){
            addition[i][j] =
matrix1[i][j] + matrix2[i][j];
        }
    }

    for (int i = 0; i < ROWS; i++){
        for (int j = 0; j < COLS; j++){
            transpose[j][i] =
matrix1[i][j];
        }
    }

    for (int i = 0; i < ROWS; i++){
        for (int j = 0; j < COLS; j++){
            multiplication[i][j] = 0;
            for (int k = 0; k < ROWS;
k++){
                multiplication[i][j] +=
matrix1[i][k] * matrix2[k][j];
            }
        }
    }

        printMatrix(addition, "Matrix
Addition");
        printMatrix(transpose,
"Transpose of Matrix 1");
        printMatrix(multiplication,
"Matrix Multiplication");

    return 0;
}
```

```
*** Matrix Addition ***
3,  4,  8,
10, 10, 8,
10, 5,  9,

*** Transpose of Matrix 1 ***
1, 5, 9,
2, 6, 4,
3, 7, 2,

*** Matrix Multiplication ***
15, 13, 28,
47, 41, 80,
40, 36, 63,
```

4.4

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.

---- funs.h ----------

```
extern void printArr(int*, int);
```

---- funs.c ----------

```
#include <stdio.h>
#include "funs.h"

void printArr(int* nums, int n){
    printf("[");
    for (int i = 0; i < n; i++){
        printf("%d, ", nums[i]);
    }
    printf("]\n");
}
```

---- main.c ----------

```
#include "funs.h"

int main(){
    int nums[] = {1, 2, 3, 4, 5, 6};
    printArr(nums, sizeof(nums)/sizeof(int));
    return 0;
}
```

Terminal

```
$ gcc -c funs.c
$ gcc -c main.c
$ gcc main.o funs.o -o a
$ ./a
[1, 2, 3, 4, 5, 6, ]
```

5.1

Implement a function which receives a pointer to a Student struct and sets the values of its fields.

```
#include <stdio.h>
struct Student{
    int id;
    char *name;
    float percentage;
};

struct Student* setStudent(struct Student *std,int id, char* name, float percentage)
{
    std->id = id;
    std->name = name;
    std->percentage = percentage;
    return std;
}

int main(){

    struct Student std;
    setStudent(&std, 10,"Ovais Ahmad", 76.8);

    printf("Id: %d\n", std.id);
    printf("Name: %s\n", std.name);
    printf("Percentage: %.2f\n", std.percentage);
    return 0;
}
```

```
Id: 10
Name: Ovais Ahmad
Percentage: 76.8
```

5.2

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

```
#include <stdio.h>

int main(int argc, char* argv[]){

    FILE *fp = fopen("w05_p2.txt", "w");
    if(fp == NULL){
        printf("File cannot be opened");
        return -1;
    }

    for (int i = 2; i < argc; i++){
        fprintf(fp, "%s\n", argv[i]);
    }
    fclose(fp);

    return 0;
}
```

commands

```
$ gcc w05_p2_cli_in_files.c
$ ./a 4 "Ovais Ahmad Khanday" "University of Kashmir" "MCA" "B-2024"
```

---

w05_p2.txt

```
Ovais Ahmad Khanday
University of Kashmir
MCA
B-2024
```

5.3

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.

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

typedef struct {
    int id;
    char name[20];
    float marks;
} Student;

int main(){

    int n = 0;
    printf("Enter number of students:
");
    scanf("%d", &n);
    Student* stds[n];
    for (int i = 0; i < n; i++){
        stds[i] = (Student
*)malloc(sizeof(Student));
        if(stds[i] == NULL){
            perror("Memory not
allocated");
            exit(1);
        }
    }
    for (int i = 0; i < n; i++){
        printf("Enter name: ");
        fflush(stdin);
        scanf("%[^\n]s", stds[i]->name);
        printf("Enter Id: ");
        scanf("%d", &stds[i]->id);
        printf("Enter Marks: ");
        scanf("%.2f", &stds[i]->marks);
    }
    for (int i = 0; i < n; i++){
        printf("%d\n", i + 1);
        printf("Id: %d\n", stds[i]->id);
        printf("Name: %s\n",
stds[i]->name);
        printf("Marks: %f\n",
stds[i]->marks);
    }

     for (int i = 0; i < n; i++){
         free(stds[i]);
    }

    return 0;
}
```

```
Enter number of students: 3
Enter name: Ovais Ahmad
Enter Id: 1
Enter Marks: 73.01

Enter name: Yawar Abass
Enter Id: 2
Enter Marks: 99.09

Enter name: John Doe
Enter Id: 43
Enter Marks: 23

1
Id: 1
Name: Ovais Ahmad
Marks: 73.01

2
Id: 2
Name: Yawar Abass
Marks: 99.09

3
Id: 43
Name: John Doe
Marks: 23.0
```

5.4

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

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

char *stuff(char *s1, char *s2, int sp, int rp)
{
    int s1Size = strlen(s1);
    int s2Size = strlen(s2);

    char *result = (char *)malloc(s1Size + s2Size + 1);
    if (!result){
        printf("Memory allocation failed.\n");
        exit(1);
    }

    int i = 0, j = 0;

    for (i = 0; i < sp; i++) result[i] = s1[i];

    for (j = 0; j < s2Size; j++, i++) result[i] = s2[j];

    for (j = sp + rp; j < s1Size; j++, i++) result[i] = s1[j];

    result[i] = '\0';

    return result;
}
int main()
{
    char *str;
    str = stuff("Hello", "World", 2, 1);
    printf("%s", str);
}
```

```
HeWorldlo
```

6.1

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

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

typedef struct{
    char name[20];
    char address[20];
    char telephone[11];
} Person;

void swap(Person* p1, Person* p2){
    Person temp;
    strcpy(temp.name, p1->name);
    strcpy(temp.address, p1->address);
    strcpy(temp.telephone,
p1->telephone);

    strcpy(p1->name, p2->name);
    strcpy(p1->address, p2->address);
    strcpy(p1->telephone,
p2->telephone);

    strcpy(p2->name,temp.name);
    strcpy(p2->address,temp.address);

strcpy(p2->telephone,temp.telephone);
}

void sort(Person p[], int n){
    // sort according to {name}
    for (int i = 0; i < n-1; i ++){
        for (int j = i + 1; j < n; j++){
            if(strcmp(p[i].name,
p[j].name) == 1){
                swap(&p[i], &p[j]);
            }
        }
    }
    // sort according to {address}
    for (int i = 0; i < n; i++)
        {
            int j = i + 1;
            if (strcmp(p[i].name,
p[j].name) == 0)
            {
                while (j < n &&
strcmp(p[i].name, p[j].name) == 0 &&
strcmp(p[j].address, p[i].address) ==
-1)
                {
                    swap(&p[i], &p[j]);
                    j++;
                }
            }
        }
}
int main(){

    int n;
    printf("Enter number of persons: ");
    scanf("%d", &n);

    Person persons[n];
    for (int i = 0; i < n;i++){
        fflush(stdin);
        printf("Enter name for (%d): ",
i + 1);
        gets(persons[i].name);
        printf("Enter address for (%d):
", i + 1);
        gets(persons[i].address);
        printf("Enter telephone for
(%d): ", i + 1);
        scanf("%10[0-9]s",
&persons[i].telephone);
    }

    sort(persons, n);
    for (int i = 0; i < n;i++){
        printf("%s, %s, %s\n",
persons[i].name, persons[i].address,
persons[i].telephone);
    }
    return 0;
}
```

```
Enter number of persons: 4

Enter name for (1): Ovais
Enter address for (1): Anantnag
Enter telephone for (1): 2323232323

Enter name for (2): Abass
Enter address for (2): Kulgam
Enter telephone for (2): 2223334456

Enter name for (3): Ovais
Enter address for (3): Aang
Enter telephone for (3): 3453453454

Enter name for (4): Abass
Enter address for (4): Anantnag
Enter telephone for (4): 22334454322

Abass, Anantnag, 2233445432
Abass, Kulgam, 2223334456
Ovais, Aang, 3453453454
Ovais, Anantnag, 2323232323
```

13.1

```
Printing Shape Details
Name: Rectangle
Color: Red
Length: 22
Width: 33
Area: 726
Perimeter: 363

Printing Shape Details
Name: Circle
Color: Blue
Radius: 43
Area: 5805.86
Perimeter: 270.04
```

```
*** Person Class ***
Name: Shahnawaz
Address: Kulgam
Age: 21

*** Person Class ***
Name: Yawar Abass
Address: Kulgam
Age: 22

*** Person Class ***
Name: Ovais Ahmad Khanday
Address: Anantnag
Age: 23

*** Employee Class ***
Name: Yawar Abass
Address: Kulgam
Age: 22
Salary: 250000
Designation: Software Engineer
Joining Date: 01/01/2024

*** Teacher Class ***
Name: Ovais Ahmad Khanday
Address: Anantnag
Age: 23
Salary: 150000
Designation: Teacher
Joining Date: 01/05/2025
Subject: ToC
Total classes in a day: 2
```

6.2

Write a program to find the number of occurrences of a word in a sentence ?

```
#include <stdio.h>
int getCount(char *str, char *ptr){
    int count = 0;
    for (int i = 0; str[i]; i++){
        int flag = 1;
        for (int j = 0; ptr[j] && str[i+j]; j++){
            if(str[i+j] != ptr[j]) {
                flag = 0;
                break;
            }
        }
        if(flag) count++;
    }
    return count;
}
int main(){

    char str[] = "She saw sea shells on the sea shore";
    int n = getCount(str, "sea");
    printf("%d", n);
    return 0;
}
```

2

6.3

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

```
#include <stdio.h>

char* concat(char* dest, char* src){
    // Given that dest can hold src
    char *currentDest = dest;
    while(*currentDest) currentDest++;
    while(*currentDest = *src){
        src++;
        currentDest++;
    }
    return dest;
}
int main(){

    char str1[100] = "Ovais";
    char str2[] = "Ahmad";
    char str3[] = "Khanday";

    concat(str1, " ");
    concat(str1, str2);
    concat(str1, " ");
    concat(str1, str3);
    printf("%s", str1);
}
```

Ovais Ahmad Khanday

6.4

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

```
#include <stdio.h>
#define FALSE 0
#define TRUE 1

int equal(char* s1, char* s2){
    while(*s1 == *s2){
        // If one string ends before another
        if((!*s1 && *s2) || (!*s2 && *s1)) return FALSE;
        // If both strings end on same length
        else if(!*s1 && !*s2) return TRUE;
        s1++;
        s2++;
    }
    return FALSE;
}

int main(){
    char s1[] = "Hello World";
    char s2[] = "Hello World";
    char s3[] = "Helo";

    printf("%d\n", equal(s1, s2));
    printf("%d\n", equal(s1, s3));
}
```

```
1
0
```

6.5

Write a program to check whether a string is a palindrome or not?

```
#include <stdio.h>
#define FALSE 0
#define TRUE 1

int isPalindrome(char str[], int n){
    int left = 0, right = n - 2;
    while(left < right){
        if(str[left] != str[right]) return FALSE;
        left++;
        right--;
    }
    return TRUE;
}

int main(){
    char str1[] = "racecar";

    char str2[] = "hello";
    printf("%d\n", isPalindrome(str1, sizeof(str1)));
    printf("%d\n", isPalindrome(str2, sizeof(str2) ));
    return 0;
}
```

```
1
0
```

6.6

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

```
#include <stdio.h>
#define FALSE 0
#define TRUE 1


int isVowel(char ch){
    char str[] = {'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U'};
    for (int i = 0; i < sizeof(str); i++){
        if(str[i] == ch) return TRUE;
    }
    return FALSE;
}
int isConsonant(char ch){
    return (((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) && (!isVowel(ch)));
}
int main(){
    char str1[100];
    printf("Enter a string: ");
    gets(str1);

    int vowels = 0, consonants = 0;

    char *ptr = str1;
    while(*ptr){
        if(isVowel(*ptr)) vowels++;
        else if(isConsonant(*ptr)) consonants++;
        ptr++;
    }

    printf("Vowels: %d\nConsonants: %d\n", vowels, consonants);

    return 0;
}
```

```
Enter a string: You were spring, And I the edge of a cliff, And a shining waterfall
rushed over me.
Vowels: 25
Consonants: 39
```

7.1

Write a program that reverse the contents of a string.

```
#include <stdio.h>

char* reverse(char str[], int n){
    for (int i = 0; i < n / 2; i++){
        char temp = str[i];
        str[i] = str[n - 1 - i];
        str[n - 1 - i] = temp;
    }
}
int main(){

    char str[] = "Hello World";
    printf("String: %s\n", str);
    reverse(str, sizeof(str)-1);
    printf("Reversed: %s\n", str);
    return 0;
}
```

```
String: Hello World
Reversed: dlroW olleH
```

7.2

Write a program to demonstrate the array indexing using pointers.

```
#include <stdio.h>
int main(){

    char *str1D = "Hello World";

    printf("%c, %c, %c, %c, %c\n", *str1D, *(str1D + 1), *(str1D + 2), *(str1D +
3),*(str1D + 4));

    char str2D[][10] = {"Hello", "World"};
    printf("%s, %s\n", *str2D, *(str2D + 1));

    printf("%c, %c, %c, %c, %c\n", *(*str2D), *(*str2D + 1), *(*str2D + 2),
*(*str2D+3), *(*str2D+4));

    printf("%c, %c, %c, %c, %c\n", *(*(str2D+1)), *(*(str2D+1) + 1), *(*(str2D+1) + 2),
*(*(str2D+1)+3), *(*(str2D+1)+4));

    return 0;
}
```

```
H, e, l, l, o
Hello, World
H, e, l, l, o
W, o, r, l, d
```

7.3

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 the structure?

```
#include <stdio.h>

typedef struct {
    int id;
    char *name;
    float marks;
} Student;

Student* fill(Student* std, int id, char* name, float marks){
    std->id = id;
    std->name = name;
    std->marks = marks;
    return std;
}
void printStd(Student* std){
    printf("Id: %d\n", std->id);
    printf("Name: %s\n", std->name);
    printf("Marks: %.2f\n", std->marks);
}

int main(){

    Student std;
    fill(&std, 10, "Ovais Ahmad", 39.01f);
    printStd(&std);
}
```

```
Id: 10
Name: Ovais Ahmad
Marks: 39.01
```

8.1

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

```
#include <stdio.h>

int main(){

    int x = 10;
    int *ptr = &x;
    int **ptrToPtr = &ptr;

    printf("x: %d\n", x);
    printf("&x: %x\n", &x);
    printf("ptr: %x\n", ptr);
    printf("*ptr: %d\n", *ptr);
    printf("&ptr: %x\n", &ptr);
    printf("ptrToPtr: %x\n", ptrToPtr);
    printf("*ptrToPtr: %x\n", *ptrToPtr);
    printf("**ptrToPtr: %d\n", **ptrToPtr);
    printf("&ptrToPtr: %x\n", &ptrToPtr);
    return 0;
}
```

```
x: 10
&x: 61fe1c
ptr: 61fe1c
*ptr: 10
&ptr: 61fe10
ptrToPtr: 61fe10
*ptrToPtr: 61fe1c
**ptrToPtr: 10
&ptrToPtr: 61fe08
```

8.2

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

```
#include <stdio.h>

int sum(int x, int y){
    return x + y;
}

int main(){

    int (*funPtr)(int, int) = sum;
    printf("Sum of 2, 4 is : %d", funPtr(2, 4));
    return 0;
}
```

```
Sum of 2, 4 is : 6
```

8.3

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

```
#include <stdio.h>
typedef struct {
    int id;
    char *name;
    float marks;
} Student;

Student* fill(Student* std, int id,
char* name, float marks){
    std->id = id;
    std->name = name;
    std->marks = marks;
    return std;
}
void printStd(Student* std){
    printf("Id: %d\n", std->id);
    printf("Name: %s\n", std->name);
    printf("Marks: %.2f\n", std->marks);
}
void swap(Student* s1, Student* s2){
    Student temp;
    temp.id = s1->id;
    temp.name = s1->name;
    temp.marks = s1->marks;

    s1->id = s2->id;
    s1->name = s2->name;
    s1->marks = s2->marks;

    s2->id = temp.id;
    s2->name = temp.name;
    s2->marks = temp.marks;
}

int main(){

    Student std1, std2;
    fill(&std1, 1, "Ovais Ahmad",
43.01f);
    fill(&std2, 12, "Yawar Abbass",
99.0f);
    swap(&std1, &std2);
    printf(" ----- Student 1 ----- \n");
    printStd(&std1);
    printf("\n ----- Student 2 -----
\n");
    printStd(&std2);
}
```

```
 ----- Student 1 -----
Id: 12
Name: Yawar Abbass
Marks: 99.00

 ----- Student 2 -----
Id: 1
Name: Ovais Ahmad
Marks: 73.01
```

8.4

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

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

class Complex
{
    float real;
    float img;

public:
    Complex(float r, float i)
    {
        real = r;
        img = i;
    }
    void print();
    void setReal(float);
    void setImg(float);
};
void Complex::print()
{
    cout << "----------------" <<
endl;
    cout << "real: " << real <<
endl;
    cout << "img: " << img <<
endl;
}
void Complex::setReal(float real)
{
    this->real = real;
}
void Complex::setImg(float img)
{
    (*this).img = img;
}

int main()
{
    Complex c1 = Complex(29, 30);
    Complex c2 = {10, 12.5f};
    Complex *c3 = new Complex(11,
22.011f);
    c1.print();
    c2.print();
    c3->print();

    c1.setReal(20.06f);
    c2.setImg(90.790f);
    c3->setReal(80.90f);
    c1.print();
    c2.print();
    c3->print();
    return 0;
}
```

```
----------------
real: 29
img: 30
----------------
real: 10
img: 12.5
----------------
real: 11
img: 22.011
----------------
real: 20.06
img: 30
----------------
real: 10
img: 90.79
----------------
real: 80.9
img: 22.011
```

8.5

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

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

class Person
{
    string name;
    int age;
    string address;
    float height;

public:
    Person(string n, int a, string ad,
float h)
    {
        name = n;
        age = a;
        address = ad;
        height = h;
    }

    void print()
    {
        cout <<
"------------------------" << endl;
        cout << "name: " << name <<
endl;
        cout << "age: " << age << endl;
        cout << "address: " << address
<< endl;
        cout << "height: " << height <<
endl;
    }
};

int main()
{
    Person p1 = {"Ovais Ahmad", 23,
"Anantnag", 5.11f};
    Person p2 = Person("Yawar", 23,
"Anantnag", 5.11f);
    p1.print();
    p2.print();
    return 0;
}
```

```
------------------------
name: Ovais Ahmad
age: 23
address: Anantnag
height: 5.11
------------------------
name: Yawar
age: 23
address: Anantnag
height: 5.11
```

9.1

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

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

class Complex
{
    float real;
    float img;

public:
    Complex(float r, float i)
    {
        real = r;
        img = i;
    }
    void print()
    {
        cout << "----------------" <<
endl;
        cout << "real: " << real <<
endl;
        cout << "img: " << img << endl;
    }
};
class Person
{
    string name;
    int age;
    string address;
    float height;

public:
    Person(string n, int a, string ad,
float h)
    {
        name = n;
        age = a;
        address = ad;
        height = h;
    }

    void print()
    {
        cout <<
"------------------------" << endl;
  cout << "name: " << name << endl;
        cout << "age: " << age << endl;
        cout << "address: " << address <<
endl;
        cout << "height: " << height << endl;
    }
};

int main()
{
    Complex c = Complex(10.0f, 12.0f);
    Person p = Person("Ovais Ahmad", 23,
"Anantnag", 5.11);

    c.print();
    p.print();

    return 0;
}
```

```
----------------
real: 10
img: 12
------------------------
name: Ovais Ahmad
age: 23
address: Anantnag
height: 5.11
```

9.2

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

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

class Complex
{
    float real;
    float img;

public:
    Complex(float r, float i)
    {
        real = r;
        img = i;
    }
    void print()
    {
        cout << "----------------" <<
endl;
        cout << "real: " << real <<
endl;
        cout << "img: " << img << endl;
    }
};
class Person
{
    string name;
    int age;
    string address;
    float height;

public:
    Person(string n, int a, string ad,
float h)
    {
        name = n;
        age = a;
        address = ad;
        height = h;
    }

    void print()
    {
        cout << "------------------------" <<
      endl;
        cout << "name: " << name << endl;
        cout << "age: " << age << endl;
        cout << "address: " << address <<
      endl;
        cout << "height: " << height << endl;
    }
};

int main()
{
    Complex c = Complex(10.0f, 12.0f);
    Person p = Person("Ovais Ahmad", 23,
      "Anantnag", 5.11);
    c.print();
    p.print();
    return 0;
}
```

```
----------------
real: 10
img: 12
------------------------
name: Ovais Ahmad
age: 23
address: Anantnag
height: 5.11
```

9.3

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

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

class Complex
{
    float real;
    float img;

public:
    Complex(float r, float i)
    {
        real = r;
        img = i;
    }
    void print();
    void setReal(float);
    void setImg(float);
    float getReal();
    float getImg();
};
void Complex::print()
{
    cout << "-----------------" << endl;
    cout << "real: " << real << endl;
    cout << "img: " << img << endl;
}
void Complex::setReal(float real)
{
    this->real = real;
}
void Complex::setImg(float img)
{
    (*this).img = img;
}
float Complex::getReal()
{
    return real;
}

float Complex::getImg()
{
    return img;
}
```

```
int main()
{
    Complex c = {10, 12.5f};
    c.print();

    c.setReal(20.06f);
    cout << "The img part is: " << c.getImg()
             << endl;
    c.print();
    return 0;
}
```

```
-----------------
real: 10
img: 12.5
The img part is: 12.5
-----------------
real: 20.06
img: 12.5
```

9.4

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

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

class Student
{
    string name;
    int age;
    float marks;

public:
    Student()
    {
        age = 18;
        marks = 33.0f;
    }
    Student(string name, int age, float
      marks)
    {
        this->name = name;
        this->age = age;
        this->marks = marks;
    }
    Student(Student &p)
    {
        this->age = p.age;
        this->name = p.name;
        this->marks = p.marks;
    }
    void print()
    {
        cout << "Name: " << name <<
      endl;
        cout << "Age: " << age << endl;
        cout << "Marks: " << marks <<
      endl;
    }
};
int main()
{

    Student s1;
      // default constructor
    Student s2 = {"Ovais Ahmad", 23, 90.0f}; //
parametrized
    Student *s3 = new Student(s2);
// copy

    s1.print();
    s2.print();
    s3->print();
    return 0;
}
```

```
Name:
Age: 18
Marks: 33
Name: Ovais Ahmad
Age: 23
Marks: 90
Name: Ovais Ahmad
Age: 23
Marks: 90
```

10.1

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

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

class Person
{

protected:
    string name;
    int age;
    string phoneNo;

public:
    Person(string name, int age, string
      phoneNo)
    {
        this->name = name;
        this->age = age;
        this->phoneNo = phoneNo;
    }
};
class Student : private Person
{
private:
    int rollNo;
    float marks;

public:
    Student(string name, int age,
      string phoneNo, int rollNo, float
      marks) : Person(name, age,
      phoneNo)
    {
        this->rollNo = rollNo;
        this->marks = marks;
    }
    void print()
    {
        cout << "Student Details" <<
      endl;
        cout << "Name: " << this->name
      << endl;
        cout << "Age: " << this->age <<
      endl;
        cout << "Phone No: " << this->phoneNo
<< endl;
        cout << "Roll No: " << this->rollNo <<
endl;
        cout << "Marks: " << this->marks <<
endl;
    }
};

int main()
{
    Student std = Student("Ovais Ahmad
Khanday", 23, "7654321234", 1, 43.01f);
    std.print();

    return 0;
    }
```

```
Student Details
Name: Ovais Ahmad Khanday
Age: 23
Phone No: 7654321234
Roll No: 1
Marks: 63.01
```

10.2

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

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

inline int square(int x) { return x * x; }

class Student
{
    string name;
    int age;

public:
    Student(string name, int age)
    {
        this->name = name;
        this->age = age;
    }
    friend int getAge(Student &);
};

int getAge(Student &std)
{
    return std.age;
}
int main()
{

    Student s = Student("ABC", 19);

    cout << getAge(s) << endl;
    cout << square(getAge(s)) << endl;

    return 0;
}
```

```
19
361
```

10.3

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

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

class Stack
{
    int *stack;
    int size;
    int capacity;

public:
    Stack()
    {
        this->capacity = 10;
        this->size = 0;
        this->stack = new
      int[this->capacity];
    }
    ~Stack()
    {
        delete[] stack;
        cout << "Destructor called" <<
      endl;
    }
    Stack(int capacity)
    {
        this->size = 0;
        this->capacity = capacity;
        this->stack = new
      int[this->capacity];
    }

    bool isEmpty()
    {
        return size == 0;
    }
    bool isFull()
    {
        return size == capacity;
    }
    bool push(int num)
    {
        if (this->isFull())
            return false;
        this->stack[size++] = num;
        return true;
    }
    int pop()
    {
        if (this->isEmpty())
            return -1;
        return this->stack[--size];
    }
    int top()
    {
        if (this->isEmpty())
            return -1;
        return this->stack[size - 1];
    }
};
int main()
{
    Stack stack = {5};
    stack.push(1);
    stack.push(2);
    stack.push(3);
    stack.push(4);
    stack.push(5);
    cout << stack.pop() << endl;
    cout << stack.pop() << endl;
    cout << stack.pop() << endl;
    cout << stack.pop() << endl;
    cout << stack.pop() << endl;

    return 0;
}
```

```
5
4
3
2
1
Destructor called
```

10.4

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

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

int sum(int x, int y)
{
    cout << "sum(int, int) called" << endl;
    return x + y;
}

float sum(float x, float y)
{
    cout << "sum(float, float) called" << endl;
    return x + y;
}

double sum(double x, double y)
{
    cout << "sum(double, double) called" << endl;
    return x + y;
}

int main()
{

    cout << sum(2, 4) << endl;
    cout << sum(232.09f, 32.32f) << endl;
    cout << sum(2032.4233233, 4.23232023) << endl;
    return 0;
}
```

```
sum(int, int) called
6
sum(float, float) called
264.41
sum(double, double) called
2036.66
```

10.5

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

```
#include <iostream>
#define ROWS 3
#define COLS 3
using namespace std;

class Matrix
{
    int matrix[ROWS][COLS];

public:
    Matrix()
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                this->matrix[i][j] = 0;
            }
        }
    }
    Matrix(int mat[ROWS][COLS])
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                this->matrix[i][j] =
mat[i][j];
            }
        }
    }

    Matrix &operator+(Matrix &m)
    {
        Matrix *r = new Matrix();
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                r->matrix[i][j] =
this->matrix[i][j] + m.matrix[i][j];
            }
        }
        return *r;
    }
    void print()
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                cout << this->matrix[i][j] <<
", ";
            }
            cout << endl;
        }
    }
};

int main()
{

    int mat1[ROWS][COLS] = {{1, 2, 3}, {2, 3,
4}, {3, 4, 5}};
    int mat2[ROWS][COLS] = {{3, 4, 5}, {1, 2,
5}, {9, 0, 1}};

    Matrix m1 = Matrix(mat1);
    Matrix m2 = Matrix(mat2);
    Matrix m3 = m1 + m2;

    m3.print();

    return 0;
}
```

```
4, 6, 8,
3, 5, 9,
12, 4, 6,
```

11.1

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

```
#include <iostream>
#define ROWS 3
#define COLS 3
using namespace std;

class Matrix
{
    int matrix[ROWS][COLS];

public:
    Matrix()
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                this->matrix[i][j] = 0;
            }
        }
    }
    Matrix(int mat[ROWS][COLS])
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                this->matrix[i][j] =
mat[i][j];
            }
        }
    }

    Matrix &operator=(Matrix &m)
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                this->matrix[i][j] =
m.matrix[i][j];
            }
        }
        return *this;
    }
    void print()
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                cout << this->matrix[i][j] <<
", ";
            }
            cout << endl;
        }
    }
};

int main()
{

    int mat1[ROWS][COLS] = {{1, 2, 3}, {2, 3,
4}, {3, 4, 5}};

    Matrix m1 = Matrix(mat1);
    Matrix m2 = m1;

    m2.print();

    return 0;
}
```

```
1, 2, 3,
2, 3, 4,
3, 4, 5,
```

11.2

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

```
#include <iostream>
#define ROWS 3
#define COLS 3
using namespace std;

class Matrix
{
    int matrix[ROWS][COLS];

public:
    Matrix()
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                this->matrix[i][j] = 0;
            }
        }
    }
    Matrix(int mat[ROWS][COLS])
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                this->matrix[i][j] = mat[i][j];
            }
        }
    }

    bool operator==(const Matrix &m)
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                if (this->matrix[i][j] !=
m.matrix[i][j])
                    return false;
            }
        }
        return true;
    }
    bool operator!=(const Matrix &m)
    {
        return !(*this == m);
    }

    void print()
    {
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLS; j++)
            {
                cout << this->matrix[i][j] << ", ";
            }
            cout << endl;
        }
    }
};

int main()
{

    int mat1[ROWS][COLS] = {{1, 2, 3}, {2, 3, 4},
{3, 4, 5}};
    int mat2[ROWS][COLS] = {{3, 4, 5}, {1, 2, 5},
{9, 0, 1}};

    Matrix m1 = Matrix(mat1);
    Matrix m2 = Matrix(mat2);
    Matrix m3 = Matrix(mat1);

    cout << "m1 == m2 : " << (m1 == m2) << endl;
    cout << "m1 == m3 : " << (m1 == m3) << endl;
    cout << "m2 == m3 : " << (m2 == m3) << endl;
    cout << "m2 == m2 : " << (m2 == m2) << endl;

    return 0;
}
```

```
m1 == m2 : 0
m1 == m3 : 1
m2 == m3 : 0
m2 == m2 : 1
```

11.3

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

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

class Student
{
    string name;
    int age;

public:
    Student(string n, int a)
    {
        name = n;
        age = a;
    }
    friend ostream &operator<<(ostream
&output, const Student &std);
    friend istream &operator>>(istream &inpu
Student &std);
    Student &operator++()
    {
        this->age++;
        return *this;
    }
};
ostream &operator<<(ostream &output, const
Student &std)
{
    output << "{ name: " << std.name << ",
age: " << std.age << " }" << endl;
    return output;
}
istream &operator>>(istream &input, Student
&std)
{
    cout << "Enter name: ";
    input >> ws;
    getline(input, std.name);
    cout << "Enter age: ";
    input >> std.age;
    return input;
}

int main()
{
    Student s = Student("John Doe", 23);
    cin >> s;
    cout << s;
    ++s;
    cout << s;
    return 0;
}
```

```
Enter name: John Doe
Enter age: 20
{ name: John Doe, age: 20 }
{ name: John Doe, age: 21 }
```

11.4

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*

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

class Person
{
protected:
    string name;
    string address;
    int age;

public:
    Person(string name, string address, int age)
    {
        this->name = name;
        this->address = address;
        this->age = age;
    }
    void print()
    {
        cout << "*** Person Class ***" << endl;
        cout << "Name: " << this->name << endl;
        cout << "Address: " << this->address << endl;
        cout << "Age: " << this->age << "\n\n";
    }
};
class Employee : protected Person
{
protected:
    double salary;
    string designation;
    string joiningDate;

public:
    Employee(string name, string address, int age, doub
salary,
             string designation, string joiningDate) :
Person(name, address, age)
    {
        this->salary = salary;
        this->designation = designation;
        this->joiningDate = joiningDate;
    }
    void print()
    {
        cout << "*** Employee Class ***" << endl;
        cout << "Name: " << this->name << endl;
        cout << "Address: " << this->address << endl;
        cout << "Age: " << this->age << endl;
        cout << "Salary: " << this->salary << endl;
        cout << "Designation: " << this->designation <<
endl;
        cout << "Joining Date: " << this->joiningDate <
"\n\n";
    }
};
class Teacher : private Employee
{
    string subject;
    int totalClassesInDay;
public:
    Teacher(string name, string address, int age, double
salary,
            string joiningDate, string subject, int
totalClassesInDay) : Employee(name, address, age, salary,
"Teacher", joiningDate)
    {
        this->subject = subject;
        this->totalClassesInDay = totalClassesInDay;
    }
    void print()
    {
        cout << "*** Teacher Class ***" << endl;
        cout << "Name: " << this->name << endl;
        cout << "Address: " << this->address << endl;
        cout << "Age: " << this->age << endl;
        cout << "Salary: " << this->salary << endl;
        cout << "Designation: " << this->designation <<
endl;
        cout << "Joining Date: " << this->joiningDate <<
endl;
        cout << "Subject: " << this->subject << endl;
        cout << "Total classes in a day: " <<
this->totalClassesInDay << "\n\n";
    }
};
int main()
{
    Person *persons[3];
    Person *p1 = new Person("Shahnawaz", "Kulgam", 21);
    Person *e1 = (Person *)new Employee("Yawar Abass",
"Kulgam", 22, 250000, "Software Engineer", "01/01/2024");
    Person *t1 = (Person *)new Teacher("Ovais Ahmad
Khanday", "Anantnag", 23, 150000, "01/05/2025", "ToC", 2);

    persons[0] = p1;
    persons[1] = e1;
    persons[2] = t1;

    persons[0]->print();
    persons[1]->print();
    persons[2]->print();

    Employee e2 = Employee("Yawar Abass", "Kulgam", 22,
250000, "Software Engineer", "01/01/2024");
    Teacher t2 = Teacher("Ovais Ahmad Khanday",
"Anantnag", 23, 150000, "01/05/2025", "ToC", 2);
    e2.print();
    t2.print();

    return 0;
}
```

12.1

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

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

class Person
{
protected:
    string name;
    int age;

public:
    Person(string name, int age)
    {
        this->name = name;
        this->age = age;
    }
    void show();
};
class Employee
{
protected:
    float salary;
    string jobDescription;

public:
    Employee(float salary, string jobDescription)
    {
        this->jobDescription = jobDescription;
        this->salary = salary;
    }
    void show();
};

class Teacher : public Person, public Employee
{
    string school;

public:
    Teacher(string name, int age, float salary, string
school) : Person(name, age), Employee(salary, "Teacher"
    {
        this->school = school;
    }
    void show();
};
void Person::show()
{
    cout << "\nPerson Class" << endl;
    cout << "Name: " << this->name << endl;
    cout << "Age: " << this->age << endl;
}
void Employee::show()
{
    cout << "\nEmployee Class" << endl;
    cout << "Salary: " << this->salary << endl;
    cout << "Job: " << this->jobDescription << endl;
}
void Teacher::show()
{
    cout << "\nTeacher Class" << endl;
    cout << "Name: " << this->name << endl;
    cout << "Age: " << this->age << endl;
    cout << "Salary: " << this->salary << endl;
    cout << "Job: " << this->jobDescription << endl;
    cout << "School: " << this->school << endl;
}

int main()
{
    Person p = Person("Robert Frost", 45);
    Employee e = Employee(334443.003f, "Writer");
    Teacher t = Teacher("John Doe", 30, 235634.0f, "ABC
School");
    p.show();
    e.show();
    t.show();
    return 0;
}
```

```
Person Class
Name: Robert Frost
Age: 45

Employee Class
Salary: 334443
Job: Writer

Teacher Class
Name: John Doe
Age: 30
Salary: 235634
Job: Teacher
School: ABC School
```

12.2

Write a program in C++ demonstrates the concept of function overriding?

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

class Person
{
protected:
    string name;
    string address;
    int age;

public:
    Person(string name, string address, int age)
    {
        this->name = name;
        this->address = address;
        this->age = age;
    }
    void print()
    {
        cout << "*** Person Class ***" << endl;
        cout << "Name: " << this->name << endl;
        cout << "Address: " << this->address <<
endl;
        cout << "Age: " << this->age << "\n\n";
    }
};
class Employee : protected Person
{
protected:
    double salary;
    string designation;
    string joiningDate;

public:
    Employee(string name, string address, int age
double salary,
             string designation, string
joiningDate) : Person(name, address, age)
    {
        this->salary = salary;
        this->designation = designation;
        this->joiningDate = joiningDate;
    }
    void print()
    {
        cout << "*** Employee Class ***" << endl;
        cout << "Name: " << this->name << endl;
        cout << "Address: " << this->address <<
endl;
        cout << "Age: " << this->age << endl;
        cout << "Salary: " << this->salary << end
        cout << "Designation: " <<
this->designation << endl;
        cout << "Joining Date: " <<
this->joiningDate << "\n\n";
    }
};

int main()
{

    Person p = Person("Ovais", "Anantnag", 23);
    Employee e = Employee("Rafaqat", "Anantnag",
23, 243000, "Teacher", "01/12/2023");

    p.print();
    e.print();
    return 0;
}
```

```
*** Person Class ***
Name: Ovais
Address: Anantnag
Age: 23

*** Employee Class ***
Name: Rafaqat
Address: Anantnag
Age: 23
Salary: 243000
Designation: Teacher
Joining Date: 01/12/2023
```

12.3

Write a C++ program to show inheritance using different levels?

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

class Person
{
public:
    string name;
    int age;

    Person(string n, int a)
    {
        name = n;
        age = a;
    }
};

class Teacher : virtual public Person
{
public:
    float salary;

    Teacher(string n, int a, float s) :
Person(n, a)
    {
        salary = s;
    }
};

class Student : virtual public Person
{
public:
    string course;
    Student(string n, int a, string c) :
Person(n, a)
    {
        course = c;
    }
};

class Researcher : public Teacher, public
Student
{
public:
    Researcher(string n, int a, float s,
string c) : Teacher(n, a, s), Student(n, a,
c), Person(n, a) {};
    friend ostream &operator<<(ostream &,
const Researcher &);
};

ostream &operator<<(ostream &output, const
Researcher &r)
{
    output << "name: " << r.name << ", age: "
<< r.age << ", course: " << r.course << ",
salary: " << r.salary << endl;
    return output;
}

int main()
{
    Researcher r = Researcher("Zaamin Gulzar",
28, 234000.0f, "English Lit");
    cout << r;
    return 0;
}
```

```
name: Zaamin Gulzar, age: 28, course: English
Lit, salary: 234000
```

12.4

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

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

class AbstractClass{
public:
    virtual void display() const = 0;
    virtual ~AbstractClass() {}
};

class ConcreteClass : public AbstractClass{
private:
    string name;

    class InnerClass{
    public:
        void printMessage() const{
            cout << "This is a message from the Inner Class.\n";
        }
    };

public:
    ConcreteClass(const string &n) : name(n) {}

    void display() const override{
        cout << "ConcreteClass name: " << name << endl;

        InnerClass inner;
        inner.printMessage();
    }
};

int main(){
    AbstractClass *obj = new ConcreteClass("ExampleClass");
    obj->display();

    delete obj;

    return 0;
}
```

```
ConcreteClass name: ExampleClass
This is a message from the Inner Class.
```

13.1

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

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

class Shape
{
    string name;
    string color;

public:
    Shape(string name, string color)
    {
        this->name = name;
        this->color = color;
    }
    virtual float getArea() = 0;
    virtual float getPerimeter() = 0;
    virtual void printShapeDetails() = 0;
    void print()
    {
        cout << "\nPrinting Shape Details" << enc
        cout << "Name: " << this->name << endl;
        cout << "Color: " << this->color << endl;
        this->printShapeDetails();
        cout << "Area: " << this->getArea() <<
endl;
        cout << "Perimeter: " <<
this->getPerimeter() << endl;
    }
    string getName()
    {
        return this->name;
    }
    string getColor()
    {
        return this->color;
    }
    void setColor(string color)
    {
        this->color = color;
    }
};

class Rectangle : public Shape
{
    float length, width;

public:
    Rectangle(string color, float length, float
width) : Shape("Rectangle", color)
    {
        this->length = length;
        this->width = width;
    }
    float getArea()
    {
        return this->length * this->width;
    }
    float getPerimeter()
    {
        return 0.5 * this->length * this->width;
    }
    void printShapeDetails()
    {
        cout << "Length: " << this->length << endl;
        cout << "Width: " << this->width << endl;
    }
};

class Circle : public Shape
{
    float radius;

public:
    Circle(string color, float radius) :
Shape("Circle", color)
    {
        this->radius = radius;
    }
    float getArea()
    {
        return 3.14 * this->radius * this->radius;
    }
    float getPerimeter()
    {
        return 3.14 * 2 * this->radius;
    }
    void printShapeDetails()
    {
        cout << "Radius: " << this->radius << endl;
    }
};

int main()
{

    Shape *rect = new Rectangle("Red", 22, 33);
    Shape *circ = new Circle("Blue", 43);

    rect->print();
    circ->print();
    return 0;
}
```

13.2

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

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

class Shape
{
public:
    virtual void draw() = 0;
    virtual ~Shape()
    {
        cout << "Shape destructor called" << endl;
    }
};
class Rectangle : public Shape
{
public:
    void draw()
    {
        cout << "Drawing rectangle" << endl;
    }
    ~Rectangle()
    {
        cout << "Rectangle destructor called" << endl;
    }
};

int main()
{
    Shape *s = new Rectangle();
    s->draw();

    delete s;
    return 0;
}
```

```
Drawing rectangle
Rectangle destructor called
Shape destructor called
```

13.3

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

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

template <class T>
void swap_(T &x, T &y)
{
    T temp = x;
    x = y;
    y = temp;
}

int main()
{
    int ia = 10, ib = 20;
    float fa = 21.05f, fb = 34.09f;
    double da = 123.31, db = 2231.322;
    char ca = 'A', cb = 'B';

    swap_<int>(ia, ib);
    swap_<float>(fa, fb);
    swap_<double>(da, db);
    swap_<char>(ca, cb);

    cout << ia << ", " << ib << endl;
    cout << fa << ", " << fb << endl;
    cout << da << ", " << db << endl;
    cout << ca << ", " << cb << endl;
}
```

```
20, 10
34.09, 21.05
2231.32, 123.31
B, A
```

13.4

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

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

template <class T>
class Calc
{
    T op1, op2;

public:
    Calc(T x, T y){
        this->op1 = x;
        this->op2 = y;
    }

    T add(){
        return this->op1 + this->op2;
    }
    T sub(){
        return this->op1 - this->op2;
    }
    T mul(){
        return this->op1 * this->op2;
    }
    T div(){
        if (this->op2 == 0)
            throw runtime_error("Division by
zero is not allowed!");
        return this->op1 / this->op2;
    }
    void print(){
        cout << "\noperands: [" << op1 << ",
<< op2 << "]" << endl;
        cout << "add: " << add() << endl;
        cout << "sub: " << sub() << endl;
        cout << "mul: " << mul() << endl;
        cout << "div: " << div() << endl;
    }
};

int main(){
    Calc<int> ic = Calc<int>(10, 20);
    Calc<float> fc = Calc<float>(20.09f,
20.32f);
    Calc<double> dc = Calc<double>(10.324,
120.431);

    ic.print();
    fc.print();
    dc.print();

    return 0;
}
```

```
operands: [10, 20]
add: 30
sub: -10
mul: 200
div: 0

operands: [20.09, 20.32]
add: 40.41
sub: -0.23
mul: 408.229
div: 0.988681

operands: [10.324, 120.431]
add: 130.755
sub: -110.107
mul: 1243.33
div: 0.0857254
```

14.1

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

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

float division(float x, float y)
{
    if (y == 0)
        throw runtime_error("Division by 0 not allowed");
    return x / y;
}

int main()
{

    try
    {
        cout << (division(12.09, 0)) << endl;
    }
    catch (const exception &e)
    {
        cerr << e.what() << '\n';
    }

    return 0;
}
```

```
Division by 0 not allowed
```

14.2

Write a C++ program to create a custom exception.

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

class MyException : public exception
{
private:
    string message;

public:
    MyException(const char *message)
    {
        this->message = message;
    }
    const char *what() const throw()
    {
        return message.c_str();
    }
};
int main()
{
    try
    {
        throw MyException("This is a custom exception");
    }
    catch (MyException &e)
    {
        cout << "Caught an exception: " << e.what() << endl;
    }
    return 0;
}
```

```
Caught an exception: This is a custom exception
```

14.3

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.

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

class Person
{
    char name[50];
    int age;
    float salary;

public:
    Person(char *name = "", int age = 0, float
salary = 0.0f)
    {
        strcpy(this->name, name);
        this->age = age;
        this->salary = salary;
    }
    friend ostream &operator<<(ostream &, const
Person &);
    Person &bonusSalary()
    {
        this->salary += (this->salary * 0.1);
        return *this;
    }
};
ostream &operator<<(ostream &output, const Persor
&person)
{
    output << "name: " << person.name << ", age:
<< person.age << ", salary: " << person.salary;
    return output;
}

class Bonus
{
    ofstream outFile;
    ifstream inFile;
    string inFileName;
    string outFileName;

public:
    Bonus(string in, string out)
    {
        this->inFileName = in;
        this->outFileName = out;
        this->inFile.open(in, ios::in |
ios::binary);
        this->outFile.open(out, ios::out |
ios::binary);
        if (!this->inFile || !this->outFile)
        {
            cerr << "Error opening files" << endl;
            exit(0);
        }
    }

    ~Bonus()
    {
        if (this->inFile.is_open())
            this->inFile.close();
        if (this->outFile.is_open())
            this->outFile.close();
    }

    Person &readPerson()
    {
        Person *p = new Person();
        inFile.read((char *)p, sizeof(Person));
        return *p;
    }

    void writePerson(Person &p)
    {
        outFile.write((char *)&p, sizeof(Person));
    }

    void generateBonus()
    {
        inFile.seekg(0, ios::beg);
        outFile.seekp(0, ios::beg);
        while (!inFile.eof())
        {
            Person p = this->readPerson();
            p.bonusSalary();
            writePerson(p);
        }
        inFile.seekg(0, ios::beg);
        outFile.seekp(0, ios::beg);
    }
```

```
};

void printFile(ifstream &fp)
{
    fp.seekg(0, ios::beg);
    while (!fp.eof())
    {
        Person p;
        fp.read((char *)&p, sizeof(Person));
        cout << p << endl;
    }
    fp.seekg(0, ios::beg);
}

int main()
{
    Person p1 = Person("William Blake", 40,
100.0f);
    Person p2 = Person("George Orwell", 57,
250.1f);
    Person p3 = Person("William Wordsworth", 60,
120.3f);

    ofstream out;
    out.open("write.dat", ios::out | ios::binary)
    out.write((char *)&p1, sizeof(Person));
    out.write((char *)&p2, sizeof(Person));
    out.write((char *)&p3, sizeof(Person));
    out.seekp(0, fstream::beg);
    out.close();

    ifstream basicFile;
    basicFile.open("write.dat");
    if (!basicFile) return -1;
    printFile(basicFile);
    basicFile.seekg(0, fstream::beg);
    basicFile.close();

    Bonus *b = new Bonus("write.dat", "bonus.dat'
    b->generateBonus();
    delete b;

    ifstream bonusFile;
    bonusFile.open("bonus.dat");
    if (!bonusFile) return -1;
    printFile(bonusFile);
    bonusFile.close();

    return 0;
}
```

```
name: William Blake, age: 40, salary: 100
name: George Orwell, age: 57, salary: 250.1
name: William Wordsworth, age: 60, salary:
120.3

name: William Blake, age: 40, salary: 110
name: George Orwell, age: 57, salary: 275.11
name: William Wordsworth, age: 60, salary:
132.33
```