**C Programming**

**Day 1**

TASK1

**1: Give 2 integers dividend and divisor,divide 2 integers without using multiply,divide and mod operator**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n1,n2;

int result=0;

printf("Enter Number:");

scanf("%d",&n1);

printf("Enter Number:");

scanf("%d",&n2);

while(n1>=n2){

n1=n1-n2;

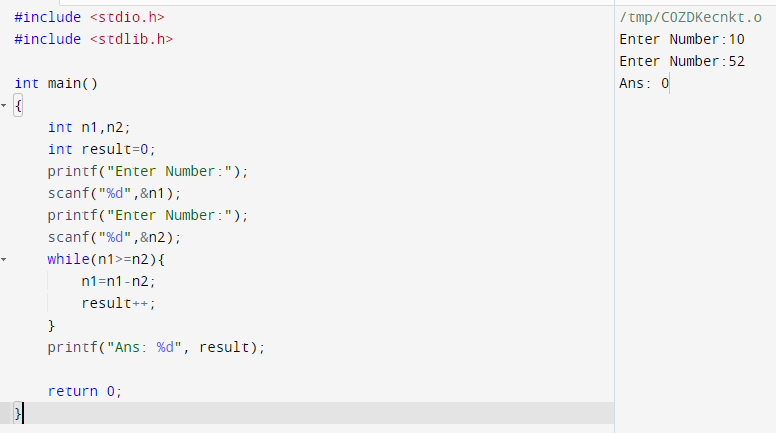
result++;

}

printf("Ans: %d", result);

return 0;

}



TASK2

**2: Palindrome no -given an int x ,return true if x is a palindrome and false otherwise**

#include <stdio.h>

#include <stdlib.h>

int main(){

int n,r,sum=0,temp;

printf("Enter Numbers: ");

scanf("%d",&n);

temp=n;

while(n>0)

{

r=n%10;

sum=(sum\*10)+r;

n=n/10;

}

if(temp==sum)

printf("The Number is palindrome number");

else

printf("The Number is Not a Palindrome number");

return 0;

}





TASK3

**3: To convert roman numbers to decimal numbers**

#include<stdio.h>

#include<string.h>

int digit(char);

int main(){

char roman\_Number[1000];

int i=0;

long int number =0;

printf("Enter the Roman number: ");

scanf("%s",roman\_Number);

while(roman\_Number[i]){

if(digit(roman\_Number[i]) < 0){

printf("Invalid roman digit : %c",roman\_Number[i]);

return 0;

}

if((strlen(roman\_Number) -i) > 2){

if(digit(roman\_Number[i]) < digit(roman\_Number[i+2])){

printf("Invalid roman number");

return 0;

}

}

if(digit(roman\_Number[i]) >= digit(roman\_Number[i+1]))

number = number + digit(roman\_Number[i]);

else{

number = number + (digit(roman\_Number[i+1]) - digit(roman\_Number[i]));

i++;

}

i++;

}

printf("Its decimal value is : %ld",number);

return 0;

}

int digit(char c){

int value=0;

switch(c){

case 'I': value = 1; break;

case 'V': value = 5; break;

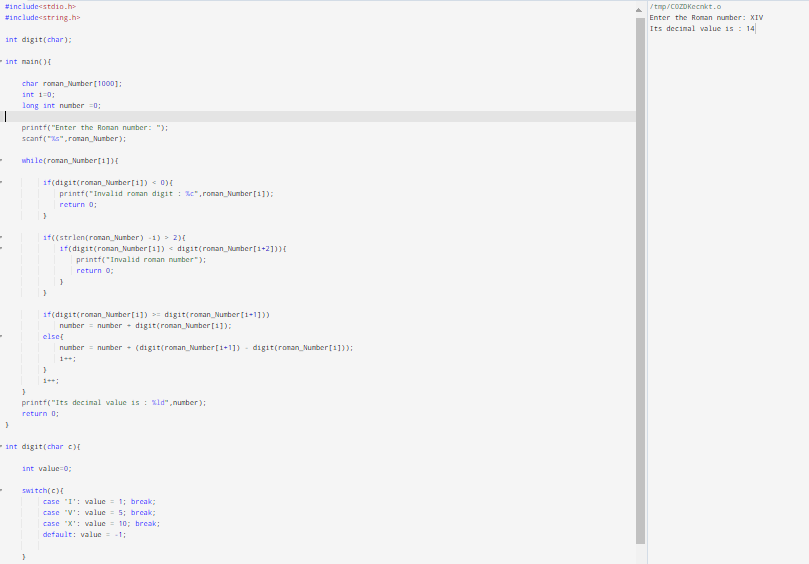
case 'X': value = 10; break;

default: value = -1;

}

return value;

}



TASK4

**4: Display auto-biography number in the given range a to b (a and b included )**

#include<stdio.h>

int check(int,int);

int getNumDigits(int);

int main(){

int no , no1;

int i ;

printf("Enter Starting Number:");

scanf("%d",&no);

printf("Enter Ending Number:");

scanf("%d",&no1);

for (i=no;i<=no1;i++){

autobio(i);

}

}

int autobio(int num)

{

int temp=0, digit=0,count=0;

temp=num;

int flag = 1;

int numDigit = getNumDigits(temp);

while(temp>0)

{

digit=temp%10;

int count=check(num,numDigit);

if(count!= digit)

{

//printf("\tNumber is not AutoBiography Number");

flag = 0;

break;

}

temp=temp/10;

numDigit--;

}

if(flag)

{

printf("%d Number is a Autobiography number\n",num);

}

}

/\* check number of times the digit appear in number \*/

int check(int num,int digit)

{

int count=0;

while(num>0)

{

if(num%10 == digit)

{

count++;

}

num=num/10;

}

return count;

}

int getNumDigits(int num)

{

//printf("%d",num);

int digits = 0;

while(num>0)

{

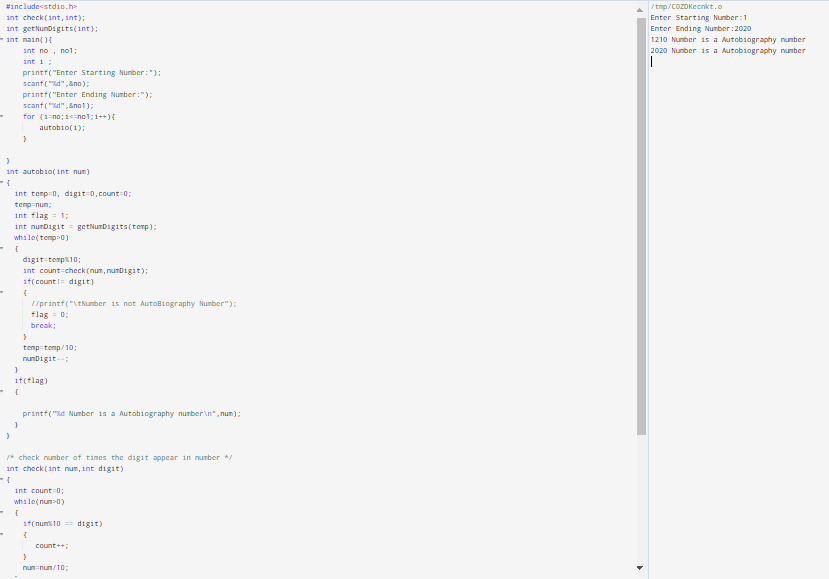
digits++;

num=num/10;

}

return digits-1;

}



TASK5

**5: Valid parameters**

**Give a strings containing just characters ‘(',’)’,’{’,’}’ determine if the input string is valid open bracket must be closed with a closed bracket**

#include<stdio.h>

int main()

{

char expression[50];

int x=0, i=0;

printf("\nEnter the Expression: ");

scanf("%s", expression);

while(expression[i]!= '\0')

{

if(expression[i]=='{')

{x++;}

if(expression[i]=='(')

{x++;}

if(expression[i]=='[')

{x++;}

else if(expression[i]=='}')

{ x--;

if(x<0)

break;

}

else if(expression[i]==')')

{ x--;

if(x<0)

break;

}

else if(expression[i]==']')

{ x--;

if(x<0)

break;

}

i++;

}

if(x==0)

{

printf("The Entered Expression %s is balanced",expression);

}

else

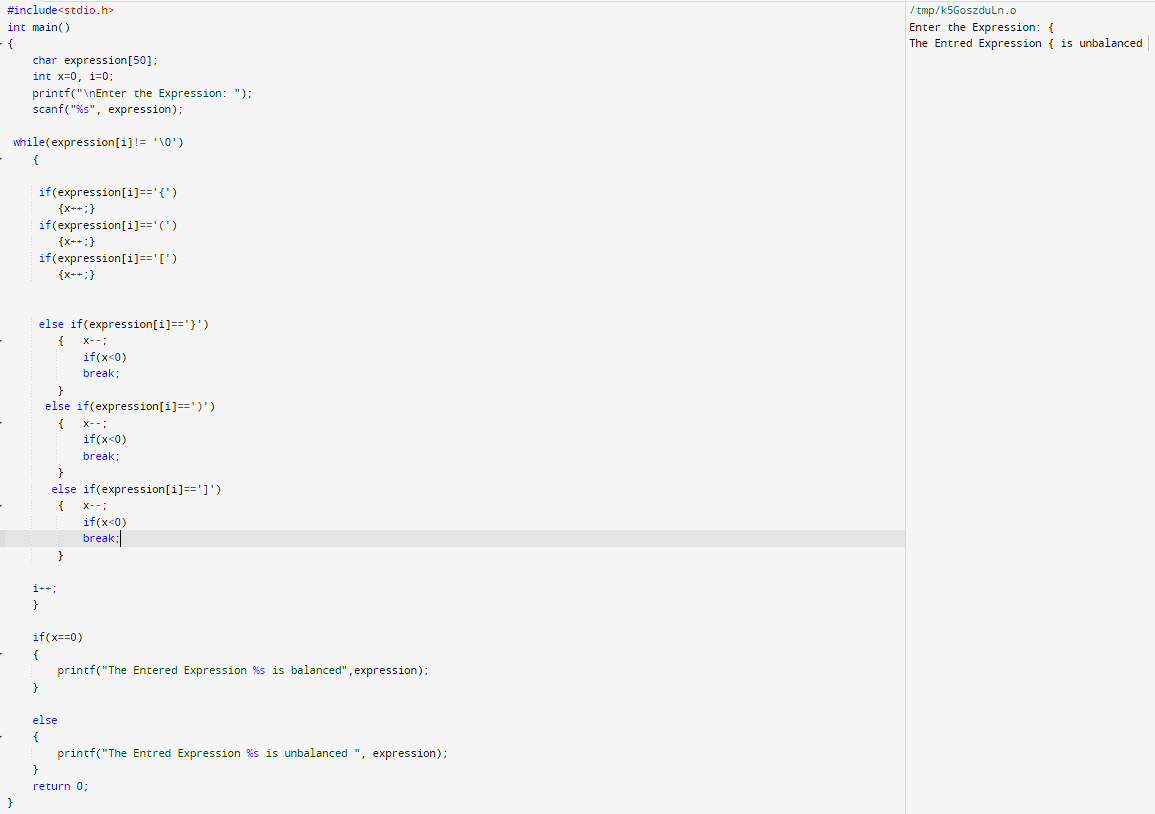
{

printf("The Entred Expression %s is unbalanced ", expression);

}

return 0;

}



**DAY 2**

TASK1

**1: Given an integer array nums and an integer val, remove all occurrences of val in nums in-place.**

**The relative order of the elements may be changed. USe function - int removeElement(int[],int)**

#include<stdio.h>

int removeElement(int\* nums, int numsSize, int val){

int count=0,temp;

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

{

if(nums[i]!=val)

{

temp=nums[i];

nums[i]=nums[count];

nums[count]=temp;

count++;

}

}

return count;

}

int main(){

//int val[5]={1,2,2,2,5};

int hell;

int Num;

int as;

printf("Enter No of inputs:");

scanf("%d",&hell);

int val[hell];

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

printf("Enter value :");

scanf("%d",&as);

val[i] = as;

}

printf("Enter No to be removed:");

scanf("%d",&Num);

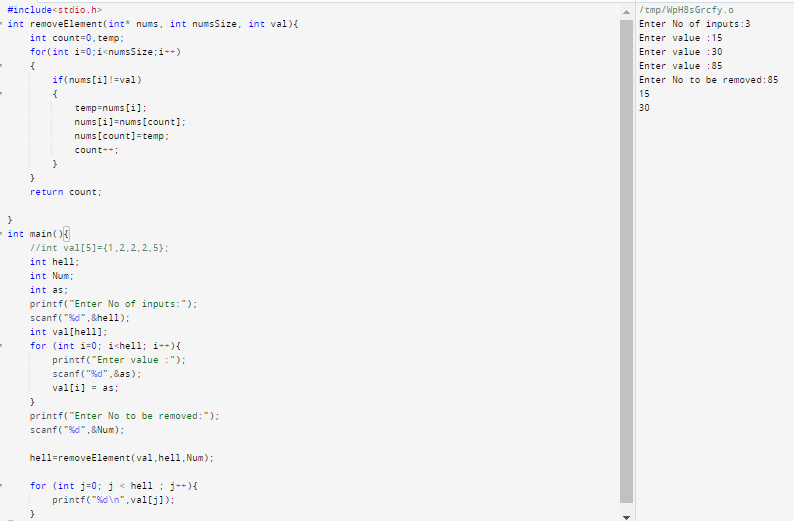
hell=removeElement(val,hell,Num);

for (int j=0; j < hell ; j++){

printf("%d\n",val[j]);

}

}



TASK2

**2:Given a sorted array of distinct integers and a target value, return the index if the target is found. If**

**not, return the index where it would be if it were inserted in order. Use function - int searchInsert**

**(int[], int)**

#include<stdio.h>

int searchInsert(int\* nums, int numsSize, int target) {

register int i;

for (i = 0; i < numsSize && target > nums[i]; i++);

return i;

}

int main(){

int hell;

int Num;

int as;

int test;

printf("Enter No of inputs:");

scanf("%d",&hell);

int val[hell];

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

printf("Enter value :");

scanf("%d",&as);

val[i] = as;

}

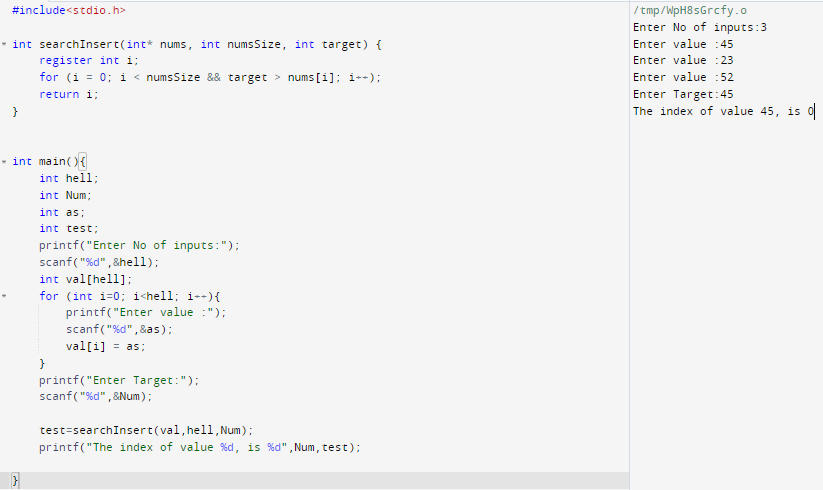
printf("Enter Target:");

scanf("%d",&Num);

test=searchInsert(val,hell,Num);

printf("The index of value %d, is %d",Num,test);

}



TASK3

**3:Given two binary strings a and b, return their sum as a binary string. Use function - char\***

**addBinary(char\*, char\*)**

#include <stdio.h>

#include <string.h>

void addBinary(char b1[], char b2[], int l1, int l2)

{

int carry = 0, temp, num1, num2, i;

char result[100];

result[l1 + 1] = '\0';

while (l2 > 0) {

num1 = b1[l1 - 1] - '0';

num2 = b2[l2 - 1] - '0';

temp = num1 + num2 + carry;

if (temp >= 2) {

carry = 1;

temp = temp % 2;

}

result[l1] = temp + '0';

l1--;

l2--;

}

while (l1 - 1 >= 0) {

temp = b1[l1 - 1] + carry - '0';

if (temp >= 2) {

carry = 1;

temp = temp % 2;

}

result[l1] = temp + '0';

l1--;

}

if (carry) {

result[0] = '1';

}

else {

for (i = 0; i < strlen(result) - 1; i++) {

result[i] = result[i + 1];

}

result[strlen(result) - 1] = '\0';

}

printf("%s + %s = %s\n", b1, b2, result);

}

int main()

{

char b1[100],b2[100];

int l1, l2;

printf("Enter binary number 1: ");

scanf("%s", b1);

printf("Enter binary number 2: ");

scanf("%s", b2);

l1 = strlen(b1);

l2 = strlen(b2);

if (l1 > l2) {

addBinary(b1, b2, l1, l2);

}

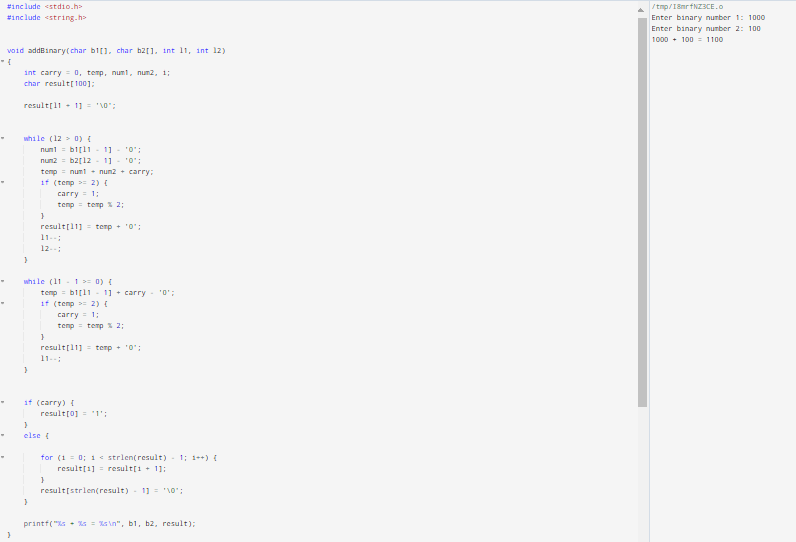
else {

addBinary(b2, b1, l2, l1);

}

return 0;

}



TASK4

**4: A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and**

**removing all non-alphanumeric characters, it reads the same forward and backward.**

**Alphanumeric characters include letters and numbers.Given a string s, return true if it is a**

**palindrome, or false otherwise. Use function - bool isPalindrome(char\*)**

#include <stdio.h>

#include <ctype.h>

#include <stdbool.h>

#include <string.h>

bool isPalindrome(char\* s) {

int len = strlen(s);

int i = 0;

int j = len - 1;

while (i < j) {

while (i < j && !isalnum(s[i])) {

i++;

}

while (i < j && !isalnum(s[j])) {

j--;

}

if (tolower(s[i]) != tolower(s[j])) {

return false;

}

i++;

j--;

}

return true;

}

int main() {

char s1[] = "A man, a plan, a canal: Panama";

char s2[] = "race a car";

printf("%s is %s\n", s1, isPalindrome(s1) ? "a palindrome" : "not a palindrome");

printf("%s is %s\n", s2, isPalindrome(s2) ? "a palindrome" : "not a palindrome");

return 0;

}



TASK5

**5: You are given an array of character letters that is sorted in non-decreasing order, and a character**

**target. There are at least two different characters in letters. Return the smallest character in letters**

**that is lexicographically greater than target. If such a character does not exist, return the first**

**character in letters. Use function - char nextGreatestLetter(char\*\*, char)**

#include <stdio.h>

char nextGreatestLetter(char\* letters, char target) {

int i = 0;

while (letters[i] <= target) {

i++;

}

return letters[i % strlen(letters)];

}

int main() {

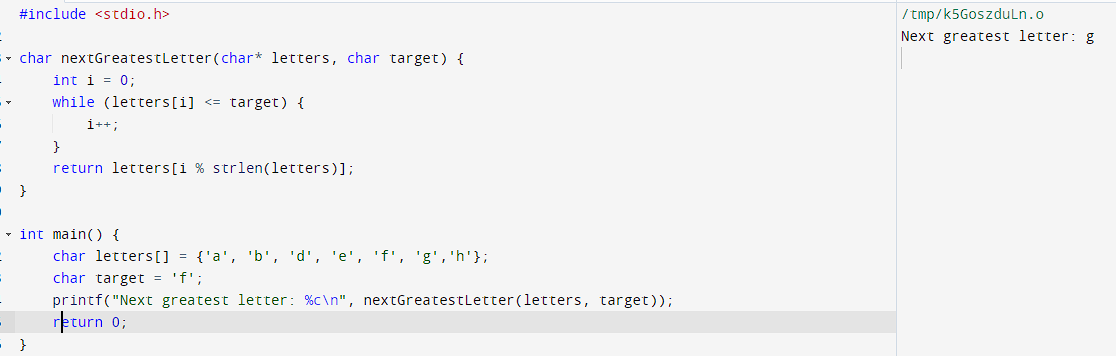
char letters[] = {'a', 'b', 'd', 'e', 'f', 'g','h'};

char target = 'f';

printf("Next greatest letter: %c\n", nextGreatestLetter(letters, target));

return 0;

}



**DAY3**

TASK1

**1:Given two strings [character pointers] s and t, return true if t is an anagram of s, and false otherwise.**

**An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase,**

**typically using all the original letters exactly once. Use a character pointer.**

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

bool isAnagram(char\* s, char\* t) {

int sLen = strlen(s);

int tLen = strlen(t);

if (sLen != tLen) {

return false;

}

int freq[26] = {0};

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

freq[s[i] - 'a']++;

}

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

freq[t[i] - 'a']--;

}

for (int i = 0; i < 26; i++) {

if (freq[i] != 0) {

return false;

}

}

return true;

}

int main() {

char s[] = "tharun";

char t[] = "anagram";

if (isAnagram(s, t)) {

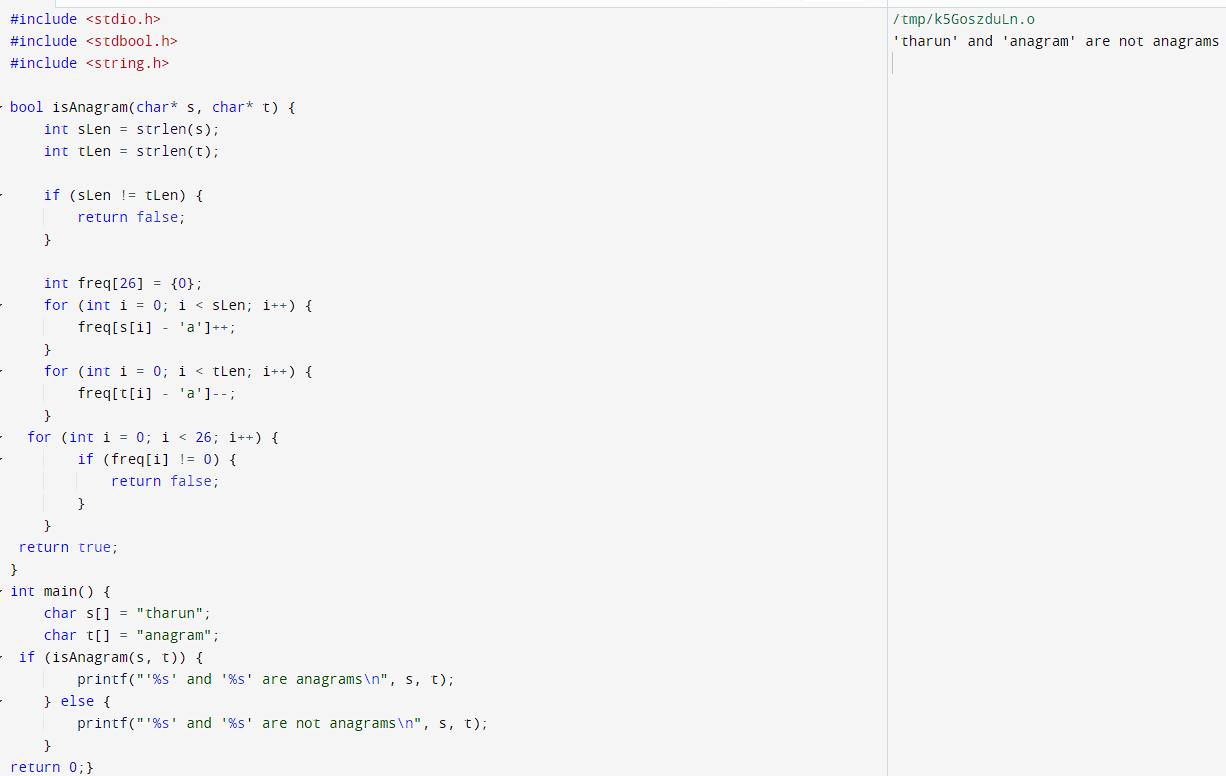
printf("'%s' and '%s' are anagrams\n", s, t);

} else {

printf("'%s' and '%s' are not anagrams\n", s, t);

}

return 0;}



TASK2

**2:You are given an integer array nums. In one move, you can pick an index i where 0 <= i < nums.**

**length and increment nums[i] by 1.**

**Return the minimum number of moves to make every value in nums unique using pointers.**

#include <stdio.h>

void bubble(int \*array, int n)

{

int i,j,temp;

for(i=0; i<n-1; i++)

{

for(j=0; j<n-i-1;j++){

if(\*(array +j)>\*(array +j+1)){

temp=\*(array+j);

}}}}

int main(){

int i,n,array[100];

printf("enter the no of array:");

scanf("%d",&n);

printf("enter %d integer:\n",n);

for(i=0;i<n;i++){

scanf("%d",array+i);

}

bubble(array,n);

printf("sorted array:\n");

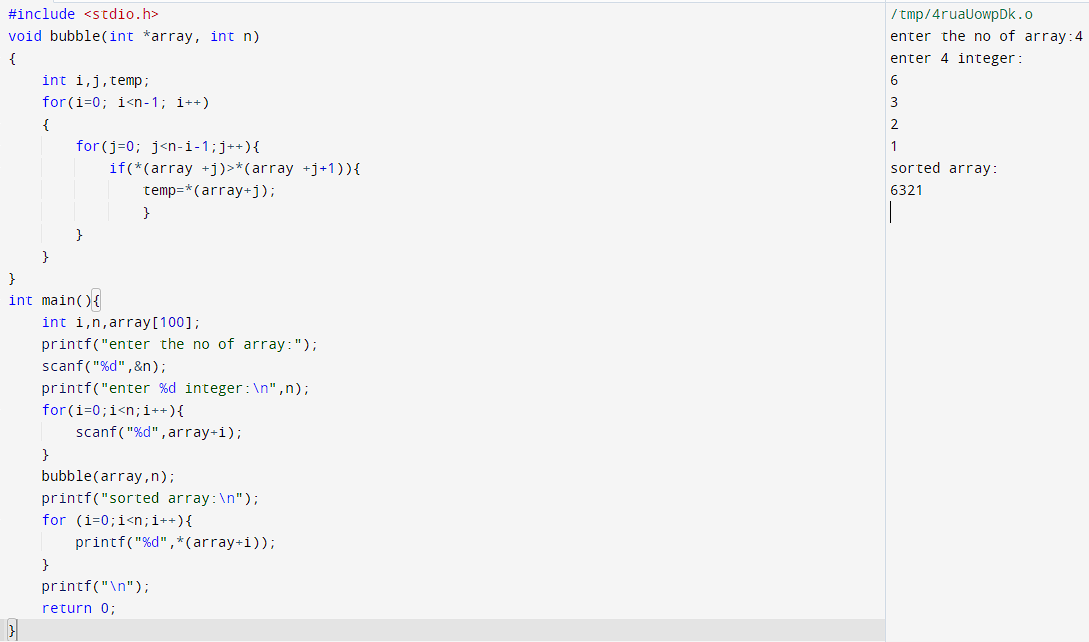
for (i=0;i<n;i++){

printf("%d",\*(array+i));

}

printf("\n");

return 0;}

****

TASK3

**3:Given a palindromic string of lowercase English letters palindrome, replace exactly one character**

**with any lowercase English letter so that the resulting string is not a palindrome and that it is the**

**lexicographically smallest one possible**

#include<stdio.h>

#include<string.h>

int main(){

char palindrome[1001];

printf("Enter String:");

scanf("%s",palindrome);

int n = strlen(palindrome);

int mid = (n-1)/2;

int i=0;

while (i < mid && palindrome[i] == palindrome[n-1-i]){

i++;

}

if (i == mid && n%2==1){

palindrome[n-1]='a';

palindrome[i]='a';

}else{

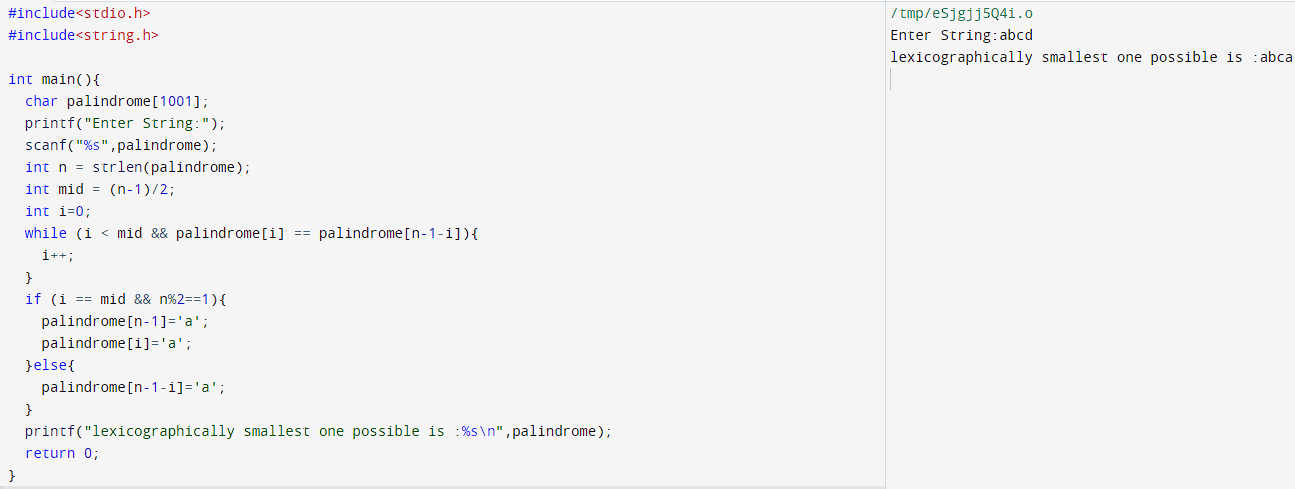
palindrome[n-1-i]='a';

}

printf("lexicographically smallest one possible is :%s\n",palindrome);

return 0;

}



TASK4

**4:Here are n children standing in a line. Each child is assigned a rating value given in the integer array ratings using pointers. You are giving candies to these children subjected to the following requirements: Each child must have at least one candy. Children with a higher rating get more candies than their neighbors. Return the minimum number of candies you need to have to distribute the candies to the children**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n,i,count,j,k,found;

int rat[100];

count=0;

found=0;

printf("Enter total number of childern:");

scanf("%d",&n);

for(i=0;i<n;i++){

printf("Enter rating of student%d:",i+1);

scanf("%d",&rat[i]);

}

for(j=1;j<n;j++){

if(rat[0]<rat[j]){

rat[0]=rat[j];

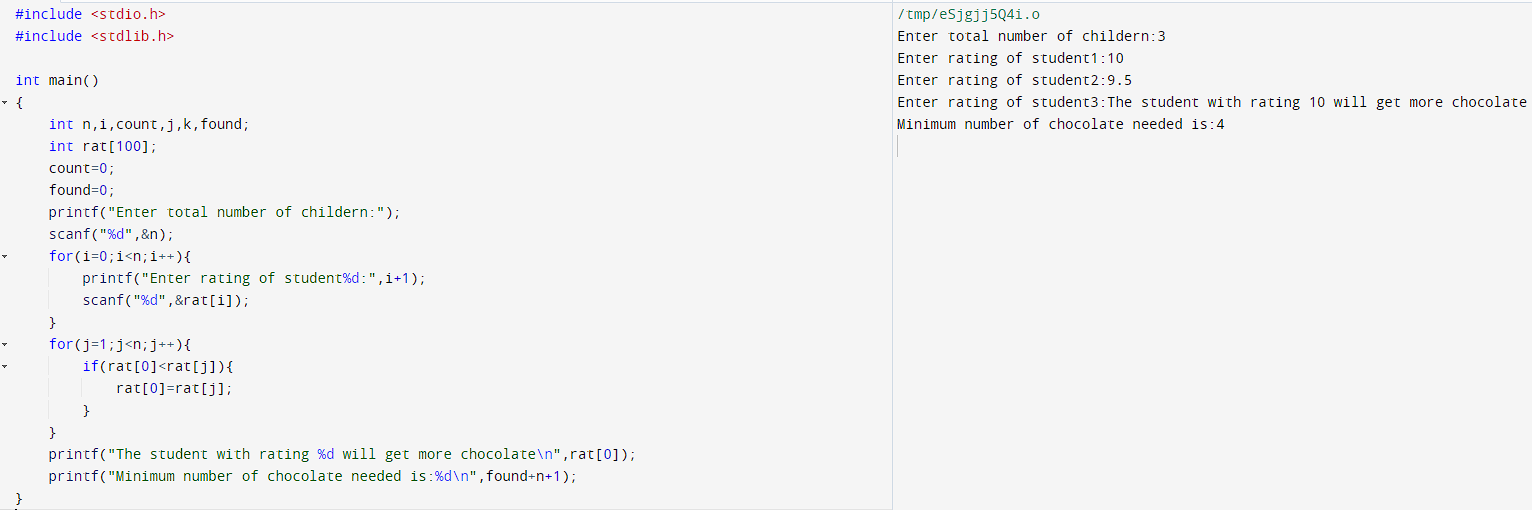
}

}

printf("The student with rating %d will get more chocolate\n",rat[0]);

printf("Minimum number of chocolate needed is:%d\n",found+n+1);

}



TASK5

**5:Sort the set of n numbers using pointers**

#include <stdio.h>

#include <stdlib.h>

void sort(int\* nums, int n) {

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

for (int j = 0; j < n-i-1; j++) {

if (\*(nums+j) > \*(nums+j+1)) {

int temp = \*(nums+j);

\*(nums+j) = \*(nums+j+1);

\*(nums+j+1) = temp; }

}

}

}

int main() {

int nums[] = {5, 2, 9, 1, 3, 6};

int n = sizeof(nums) / sizeof(nums[0]);

sort(nums, n);

printf("Sorted array: ");

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

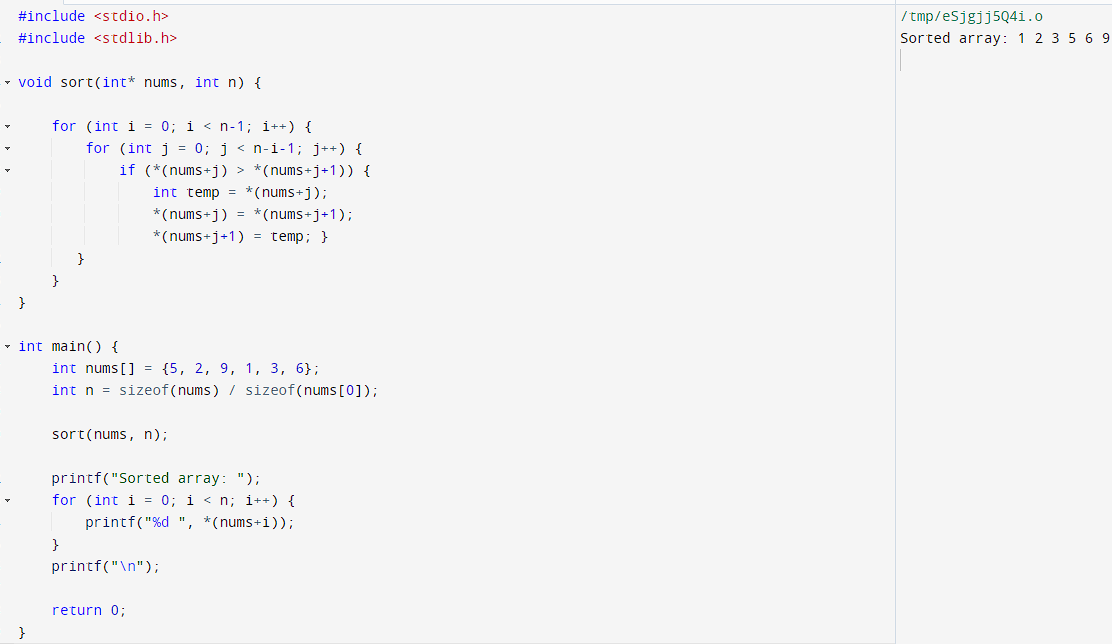
printf("%d ", \*(nums+i));

}

printf("\n");

return 0;

}

****

**DAY 4&5**

TASK1

**1: Write a short C program that declares and initializes (to any value you like) a double, an int, and a char. Next declare and initialize a pointer to each of the three variables. Your program should then print the address of, and value stored in, and the memory size (in bytes) of each of the six variables. Use the “0x%x” formatting specifier to print addresses in hexadecimal. You should see addresses that look something like this: "0xbfe55918". The initial characters "0x" tell you that hexadecimal notation is being used; the remainder of the digits give the address**

**itself. Use “%f” to print a floating value. Use the sizeof operator to determine the memory size allocated for each variable.**

#include <stdio.h>

int main() {

double myDouble = 3.14159;

int myInt = 42;

char myChar = 'A';

double \*doublePtr = &myDouble;

int \*intPtr = &myInt;

char \*charPtr = &myChar;

printf("Address of myDouble: 0x%x\n", (unsigned int)&myDouble);

printf("Value of myDouble: %f\n", myDouble);

printf("Memory size of myDouble: %lu bytes\n\n", sizeof(myDouble));

printf("Address of myInt: 0x%x\n", (unsigned int)&myInt);

printf("Value of myInt: %d\n", myInt);

printf("Memory size of myInt: %lu bytes\n\n", sizeof(myInt));

printf("Address of myChar: 0x%x\n", (unsigned int)&myChar);

printf("Value of myChar: %c\n", myChar);

printf("Memory size of myChar: %lu bytes\n\n", sizeof(myChar));

printf("Address of doublePtr: 0x%x\n", (unsigned int)&doublePtr);

printf("Value of doublePtr: 0x%x\n", (unsigned int)doublePtr);

printf("Memory size of doublePtr: %lu bytes\n\n", sizeof(doublePtr));

printf("Address of intPtr: 0x%x\n", (unsigned int)&intPtr);

printf("Value of intPtr: 0x%x\n", (unsigned int)intPtr);

printf("Memory size of intPtr: %lu bytes\n\n", sizeof(intPtr));

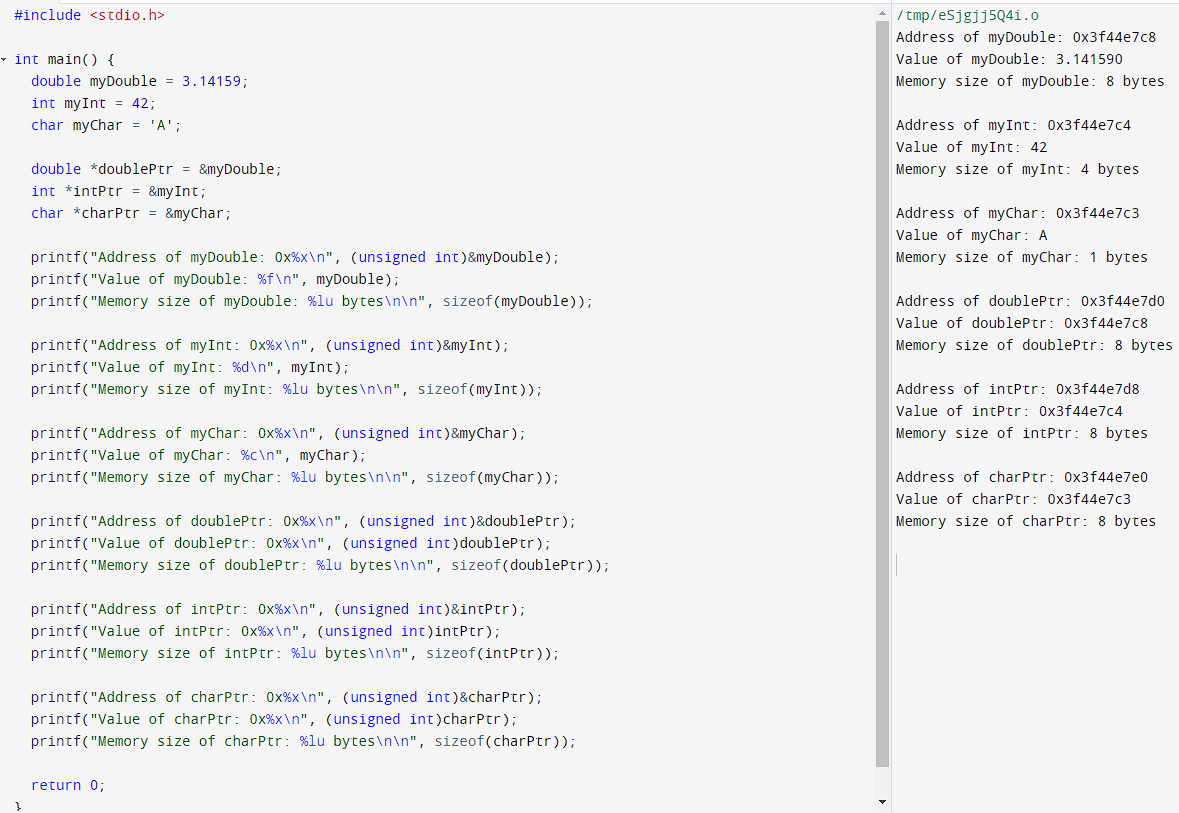
printf("Address of charPtr: 0x%x\n", (unsigned int)&charPtr);

printf("Value of charPtr: 0x%x\n", (unsigned int)charPtr);

printf("Memory size of charPtr: %lu bytes\n\n", sizeof(charPtr));

return 0;

}



TASK2

**2: Implement a program that differentiate between ordinary arguments, which are passed by value, and pointer arguments, which are passed by reference.**

#include <stdio.h>

void swap(int x ,int y )

{

int temp=x;

x=y;

y=temp;

}

void swap1(int \*x,int \*y)

{

int temp = \*x;

\*y = temp;

}

int main()

{

int x=10;

int y=11;

//call by value//

printf(" value before swap : x=%d, y= %d\n",x,y);

swap(x,y);

printf("Value after swap:x=%d , y=%d \n",x,y);

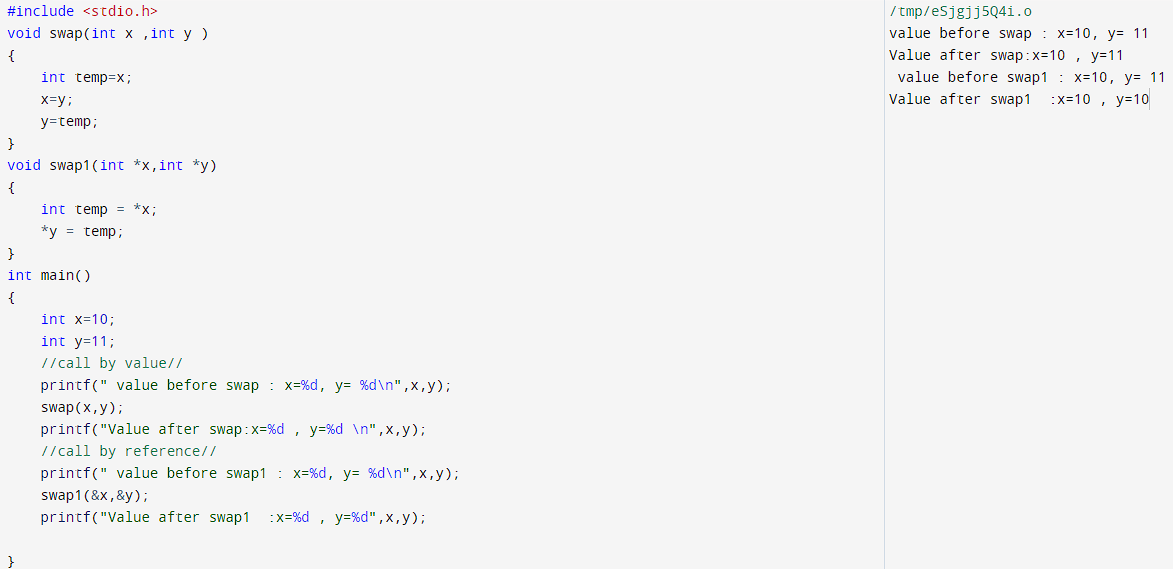
//call by reference//

printf(" value before swap1 : x=%d, y= %d\n",x,y);

swap1(&x,&y);

printf("Value after swap1 :x=%d , y=%d",x,y);

}



TASK3

**3:Count the number of vowels, consonants, digits, whitespace characters and “other” characters (punctuation, operators, brackets, etc.). Read a line of text, store it in a one-dimensional character array, and then analyze the individual array elements. An appropriate counter will be incremented for each character. The value of each counter (number of vowels, number of consonants, etc.) can then be written out after all of the characters have been analyzed**

#include <stdio.h>

#include <ctype.h>

int main() {

char str[100];

int vowels = 0, consonants = 0, digits = 0, whitespace = 0, other = 0;

int i;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

for (i = 0; str[i] != '\0'; i++) {

if (isalpha(str[i])) {

switch (tolower(str[i])) {

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

vowels++;

break;

default:

consonants++;

break;

}

} else if (isdigit(str[i])) {

digits++;

} else if (isspace(str[i])) {

whitespace++;

} else {

other++;

}

}

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

printf("Consonants: %d\n", consonants);

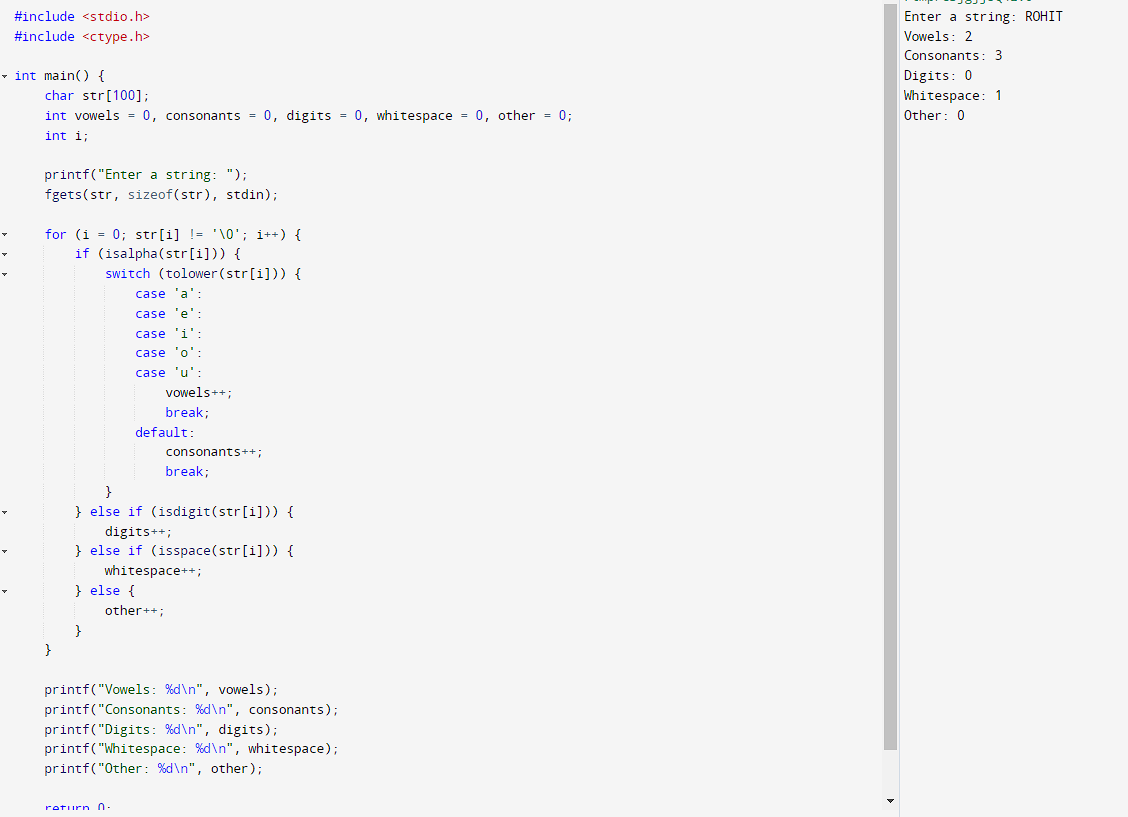
printf("Digits: %d\n", digits);

printf("Whitespace: %d\n", whitespace);

printf("Other: %d\n", other);

return 0;

}



**Assignment1:**

**Develop a program that will accept three integer quantities, indicating the month, day and year, and then display the corresponding day of the week, the month, the day and the year in a more legible manner. For example, suppose we were to enter the date 5 24 1997; this would produce the output Saturday, May 24, 1997 The computation can be carried out using the following empirical rules. 1. Enter numerical values for the variables mm, dd and yy, which represent the month, day and year, respectively (e.g., 5 24 1997). 2. Determine the approximate day of the current year, as ndays = (long) (30.42 \* (mm - 1)) + dd; 3. If mm == 2 (February), increase the value of ndays by 1. 4. If mm > 2 and mm < 8 (March, April, May, June or July), decrease the value of ndays by 1. 5. Convert the year into the number of years beyond the base date; i.e., yy - = 1900. Then test for a leap year as follows: If ( yy % 4) == 0 and mm > 2, increase the value of ndays by 1. 6. Determine the number of complete 4-year cycles beyond the base date as yy / 4. For each complete 4-year cycle, add 1461 to ndays. 7. Determine the number of full years beyond the last complete 4-year cycle as yy % 4. For each full year, add 365 to ndays. Then add 1, because the first year beyond a full 4-year cycle will be a leap year. 1 3 12 POINTERS 8. If ndays > 59 (i.e., if the date is any day beyond February 28, 1900), decrease the value of ndays by 1, because 1900 is not a leap year. (Note that the last year of each century is not a leap year, except those years that are evenly divisible by 400. Therefore 1900, the last year of the ninteenth century, is not a leap year, but 2000, the last year of the twentieth century, is a leap year.) 9. Determine the numerical day of the week corresponding to the specified date as day = (ndays % 7). Note that day == 1 corresponds either to the base date, which is a Monday, or another date that also occurs on a Monday. Hence, day == 2 will refer to a Tuesday, day == 3 will refer to a Wednesday, . . .,day == 6 will refer to a Saturday, and day == 0 will refer to a Sunday**

#include <stdio.h>

int main()

{

int mm, dd, yy, ndays, day, leap\_years;

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

const char \*days[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

printf("Enter date (dd mm yyyy): ");

scanf("%d %d %d", &dd, &mm, &yy);

ndays = (long)(30.42 \* (mm - 1)) + dd;

if (mm == 2 && yy % 4 == 0) // Leap year

ndays++;

else if (mm > 2 && mm < 8)

ndays--;

yy -= 1900;

leap\_years = yy / 4;

ndays += leap\_years \* 1461;

ndays += (yy % 4) \* 365;

if (yy % 4 > 0)

ndays++;

if (ndays > 59) // February 28th

ndays--;

day = ndays % 7;

printf("%s, %s %d, %d\n", days[day], months[mm - 1], dd, yy + 1900);

return 0;

}



**Assignment2: Sort the names of the students in a class using the function and return the number of swaps needed. Use function- int sort(\*\*char). If first character is same then check using the second character. If there two same names then their relative order should not be changed**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int sort(char\*\* names, int n);

int main() {

int n;

printf("Enter the number of students: ");

scanf("%d", &n);

char\*\* names = (char\*\*)malloc(n \* sizeof(char\*));

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

printf("Enter the names of %d students:",i);

names[i] = (char\*)malloc(50 \* sizeof(char));

scanf("%s", names[i]);

}

int swaps = sort(names, n);

printf("The sorted names are:\n");

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

printf("%s\n", names[i]);

}

printf("Number of swaps needed: %d\n", swaps);

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

free(names[i]);}

free(names);

return 0;}

int sort(char\*\* names, int n) {

int swaps = 0;

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

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

if (strcmp(names[i], names[j]) > 0) {

// swap names[i] and names[j]

char\* temp = names[i];

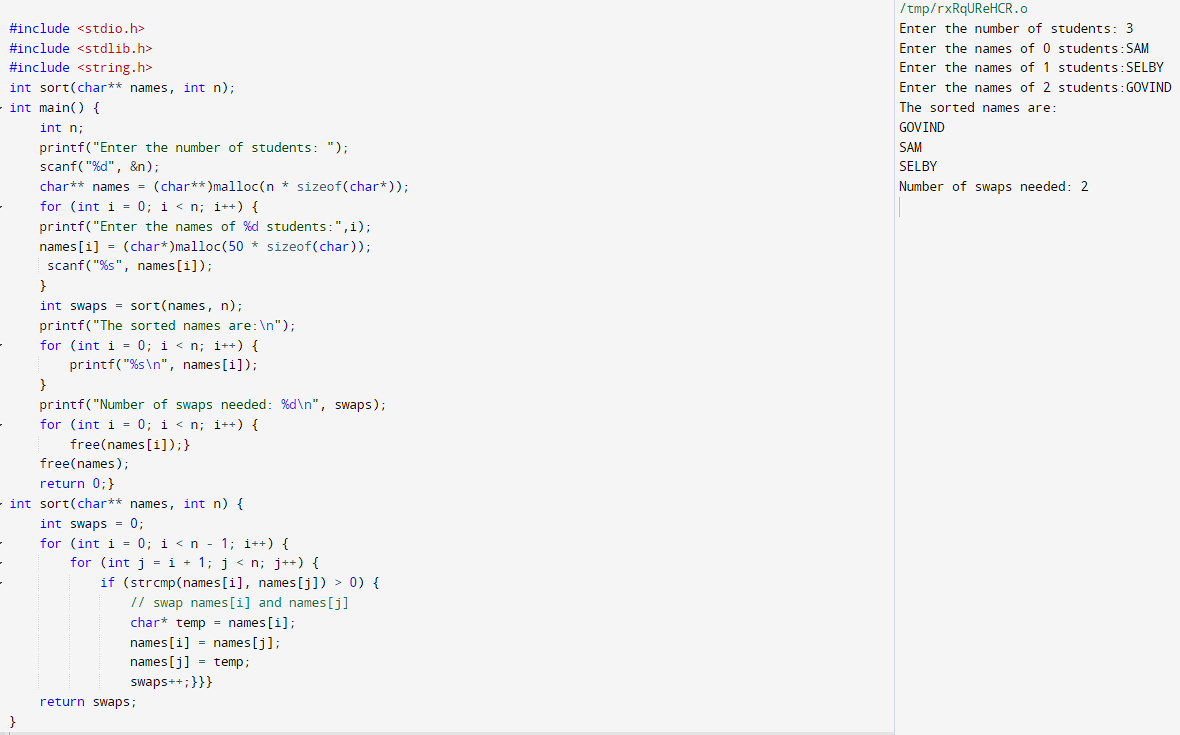
names[i] = names[j];

names[j] = temp;

swaps++;}}}

return swaps;

}



**DAY6**

**FN1 TASK**

**1.Implement the two standard functions of Dynamic Memory Allocation malloc() and calloc()**

#include<stdio.h>

#include<stdlib.h>

int mallocate(){

int num;

printf("Enter No of Memory Block Needed:");

scanf("%d",&num);

int \*ptrm = (int \*)malloc(num\*sizeof(int));

if (ptrm==NULL){

printf("Memory is FUll!");}

else{

printf("Memory is Sucessfully allocated and Total Memory allocated is : %zu bytes\n",sizeof(\*ptrm));}}

int callocate(){

int num;

int size;

printf("Enter No of Memory Block Needed:");

scanf("%d",&num);

printf("Enter size of Each Memory Block:");

scanf("%d",&size);

int \*ptrc = (int \*)calloc(num,size);

if (ptrc==NULL){

printf("Memory is Not Available");}

else{

printf("Memoery is successfully allocated and Total Memory allocated is : %zu bytes\n",sizeof(\*ptrc));}}

int main(){

int j=0;

int catch;

while (j<1){

printf("1.malloc\n2.calloc\n-->");

scanf("%d",&catch);

if (catch==1){

mallocate();

}else if(catch==2){

callocate();}

else{

break;}}}



**FN2 TASK**

**2.Implement the two standard functions of Dynamic Memory Allocation realloc() and free()**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main() {

int size = 5;

char \*str1, \*str2;

str1 = (char\*) malloc(size \* sizeof(char));

if (str1 == NULL) {

printf("Memory allocation failed!\n");

return 1;}

strcpy(str1, "hello");

printf("String 1: %s\n", str1);

str1 = (char\*) realloc(str1, 10 \* sizeof(char));

if (str1 == NULL) {

printf("Memory reallocation failed!\n");

return 1;}

strcat(str1, " world");

printf("String 1: %s\n", str1);

str2 = (char\*) calloc(size, sizeof(char));

if (str2 == NULL) {

printf("Memory allocation failed!\n");

free(str1);

return 1;}

strcpy(str2, "welcome");

printf("String 2: %s\n", str2);

free(str1);

free(str2);

return 0;

}



**DAY7**

**Storage Class Local Variable**

#include <stdio.h>

int main(void)

{

int i;

for (int i = 0; i < 5; ++i) {

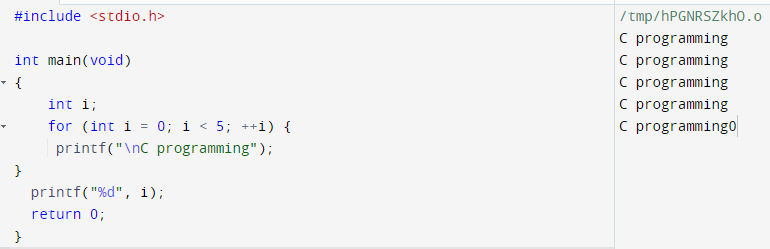
printf("\nC programming");

}

printf("%d", i);

return 0;

}

****

**Global Variable**

#include <stdio.h>

void display();

int n = 5;

int main()

{

++n;

display();

return 0;

}

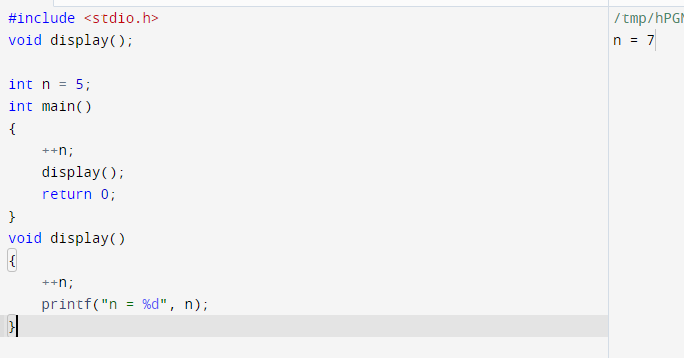
void display()

{

++n;

printf("n = %d", n);

}



**Static Variable**

#include <stdio.h>

void display();

int main()

{

display();

display();

display();

display();

}

void display()

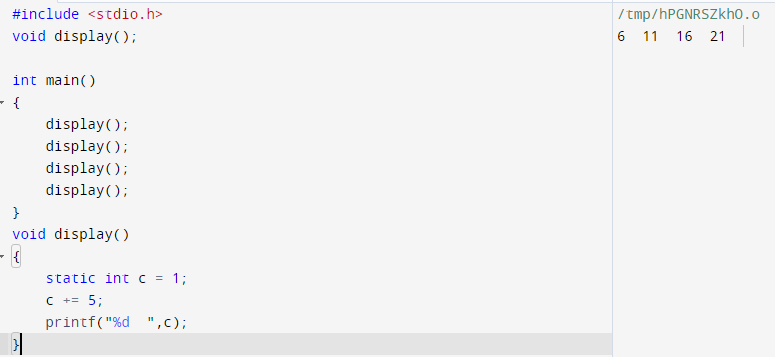
{

static int c = 1;

c += 5;

printf("%d ",c);

}



TASK1

**FN1 task: Consider the C code given below and predict the output with auto and static storage classes for the variable num and a.**

**A)**

#include <stdio.h>

int fun()

{

static int num = 16;

return num--;

}

int main()

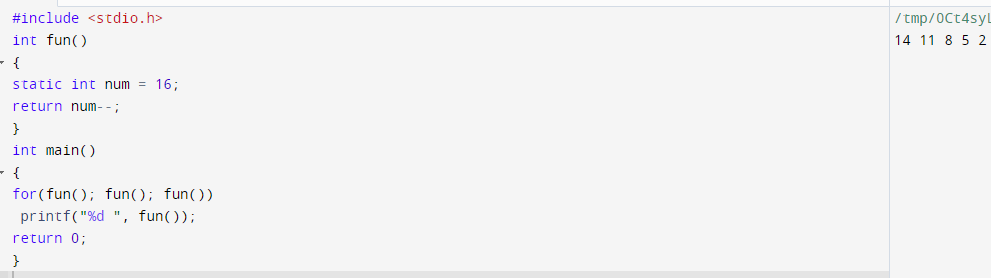
{

for(fun(); fun(); fun())

printf("%d ", fun());

return 0;

}



**B)**

#include<stdio.h>

int func(){

int a=0;

int b=0;

a++;

b++;

printf(“a= %d and b= %d\n”, a, b);

}

int main() {

int a = 10;

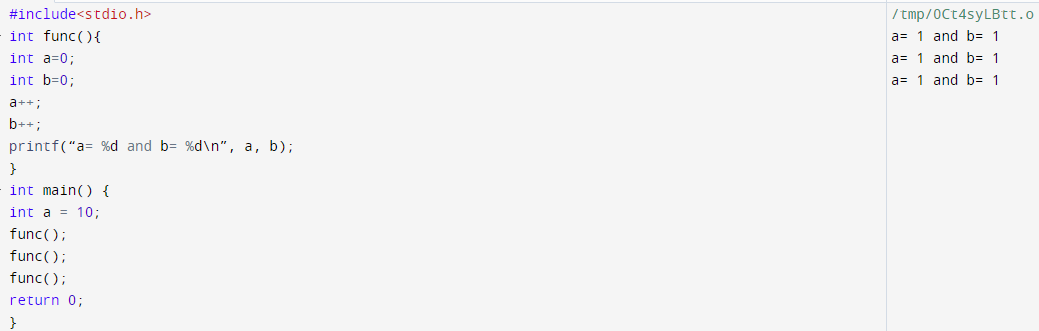
func();

func();

func();

return 0;

}



**DAY8**

**FN1 task:**

**Declare a FILE type pointer variable to store reference of file, say FILE \* fPtr = NULL;. Create or open file using fopen() function. fopen() function is used to open a file in different mode. You can open a file in basic three different mode r(read), w(write) and a(append) mode. We will use w file mode to create a file. fopen("file-name", "read-mode"); function accepts two parameter first file name to read/create/write/append data, next is file open mode. On success it return pointer to FILE type otherwise NULL pointer. Input data from user to write into file, store it to some variable say data. C provides several functions to perform IO operation on file. For this post to make things simple I will use fputs() function to write data to file. fputs("content-to-write", stream) function accepts two parameters. First string data to write into file, next pointer to FILE type that specifies where to write data. Use fputs() function to write data to fPtr i.e. perform fputs(data, fPtr);. Finally after completing all operations you must close file, to save data written on file. Use fclose (fPtr) function to close file**

#include <stdio.h>

int main() {

FILE \*fPtr = NULL;

char data[50];

fPtr = fopen("example1.txt", "w");

if (fPtr == NULL) {

printf("Error opening file.\n");

return 1;

}

printf("Enter data to write to file: ");

fgets(data, sizeof(data), stdin);

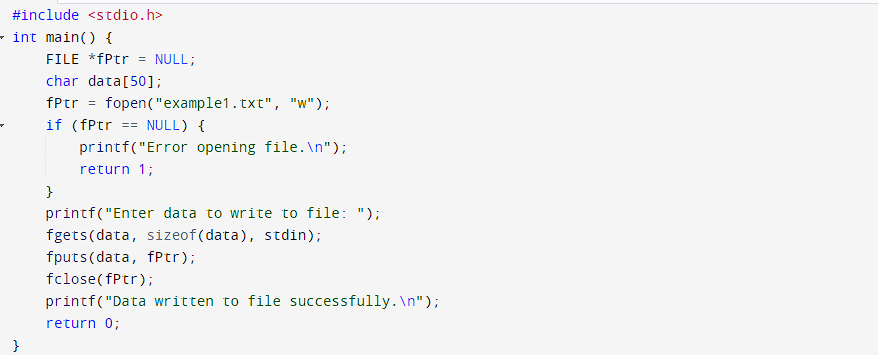
fputs(data, fPtr);

fclose(fPtr);

printf("Data written to file successfully.\n");

return 0;

}



**FN2 task: Implement the availability of a file and implement it using fopen(), access() and stat() function.**

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

int main() {

char \*filename = "example.txt";

if (access(filename, R\_OK) != 0) {

printf("File %s is not available for reading.\n", filename);

return 1;}

struct stat st;

if (stat(filename, &st) != 0) {

printf("Error getting file stats for %s.\n", filename);

return 1;}

if (st.st\_size == 0) {

printf("File %s is empty.\n", filename);

} else {

printf("File %s is available and has size %ld bytes.\n", filename, st.st\_size);}

FILE \*fp = fopen(filename, "r");

if (fp == NULL) {

printf("Error opening file %s.\n", filename);

return 1;}

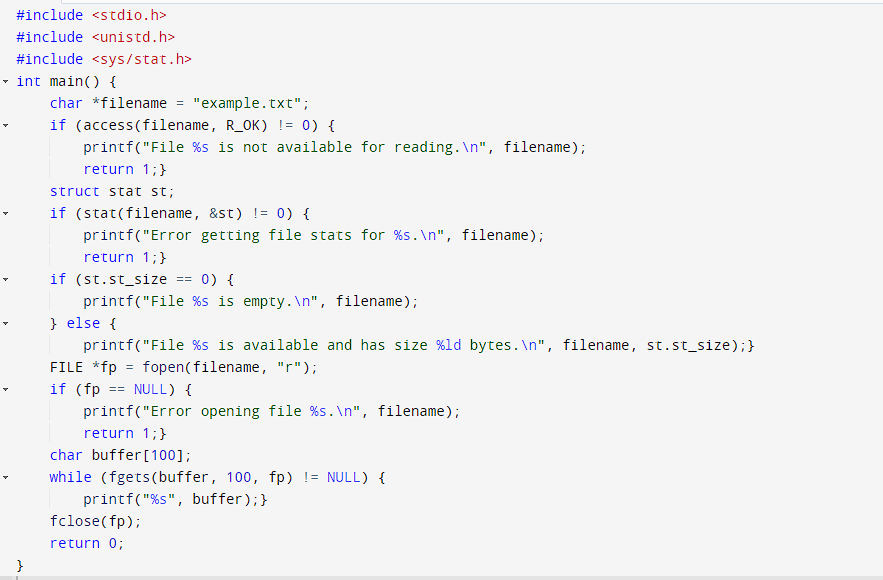
char buffer[100];

while (fgets(buffer, 100, fp) != NULL) {

printf("%s", buffer);}

fclose(fp);

return 0;}



**AN1 task:**

**list all files and directories of a directory**

**Input source path to list all files and sub-directories. Store it in some variable say path.**

**Open directory stream using opendir() and store its reference to \*dir of DIR type.**

**Initialize another variable of pointer to structure dirent type, say struct dirent \*dp.**

**Read next element from directory stream using dp = readdir(dir).**

**Print current directory stream item name, using dp->name.**

**Repeat step 4-5 till dp != NULL.**

**Finally, close the directory stream pointed by dir variable.**

**replace specific line in a text file**

**Open source file in read mode, store its reference to fPtr.**

**Create and open a temporary file with name replace.tmp, store its reference to fTemp.**

**Input line number to replace in file from user. Store it in some variable say line.**

**Input new line from user to replace with, store it in newline.**

**Initialize a count variable with 0.**

**Read a line from file and store it in buffer.**

**Increment count by 1.**

**If count == line, then current line should be replaced with newline. Means if (count == 0) then write**

**newline to fTemp, otherwise write buffer to fTemp.**

**Repeat step 6-8 till end of file.**

**Finally close all files.**

**Delete the original source file and rename temporary fTemp file path as of source file.**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define BUFFER\_SIZE 1024

int main() {

char\* filename = "/user/desktop/example.txt ";

FILE\* fPtr = fopen(filename, "r");

if (fPtr == NULL) {

printf("Error: Failed to open file.\n");

return 1;

}

char\* tempFilename = "replace.tmp";

FILE\* fTemp = fopen(tempFilename, "w");

if (fTemp == NULL) {

printf("Error: Failed to create temporary file.\n");

fclose(fPtr);

return 1;

}

int line;

printf("Enter line number to replace: ");

scanf("%d", &line);

char newline[BUFFER\_SIZE];

printf("Enter new line: ");

fgets(newline, BUFFER\_SIZE, stdin);

fgets(newline, BUFFER\_SIZE, stdin);

int count = 0;

char buffer[BUFFER\_SIZE];

while (fgets(buffer, BUFFER\_SIZE, fPtr) != NULL) {

count++;

if (count == line) {

fputs(newline, fTemp);

} else {

fputs(buffer, fTemp);

}

}

fclose(fPtr);

fclose(fTemp);

remove(filename);

rename(tempFilename, filename);

return 0;

}



**DAY9**

**C++**

**FN1 task:**

**Object and Class in C++ Programming.**

**Using Public and Private in C++ Class.**

**Develop simple programs like calculating the area of a circle using the above 2 tasks.**

#include <iostream>

using namespace std;

class Circle {

public:

void setRadius(double r);

double getArea();

private:

double radius;

const double pi = 3.14159;

};

void Circle::setRadius(double r) {

radius = r;

}

double Circle::getArea() {

return pi \* radius \* radius;

}

int main() {

Circle myCircle;

double radius;

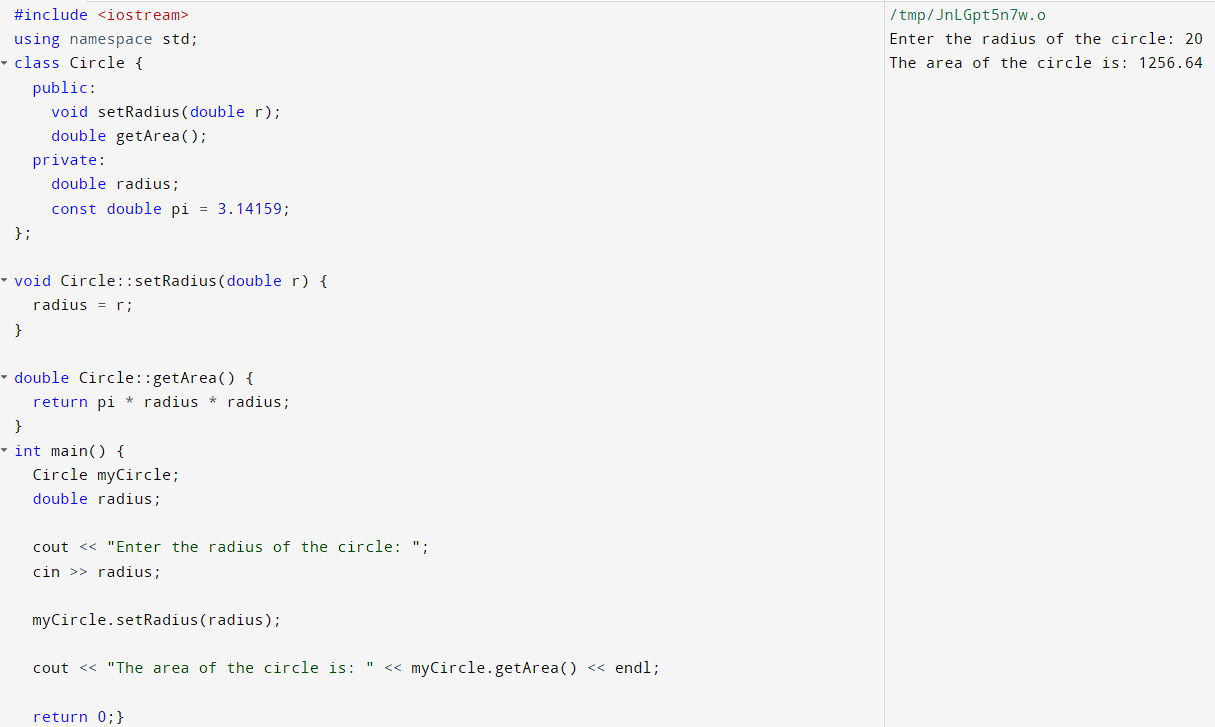
cout << "Enter the radius of the circle: ";

cin >> radius;

myCircle.setRadius(radius);

cout << "The area of the circle is: " << myCircle.getArea() << endl;

return 0;}



**DAY 10**

**FN1 Task:**

**create classes for Bus, Car, and Truck. The methods fuelAmount(), capacity(), applyBrakes() will be the same for all three classes**

#include <iostream>

class Bus {

public:

int fuelAmount() { return 4500; }

int capacity() { return 50; }

int applybreak(){ return 5 ;}

};

class Car {

public:

int fuelAmount() { return 3500; }

int capacity() { return 5; }

int applybreak(){ return 15 ;}

};

class Truck {

public:

int fuelAmount() { return 6000; }

int capacity() { return 3; }

int applybreak(){ return 10 ;}

};

int main() {

Bus bus;

Car car;

Truck truck;

std::cout << "Bus fuel amount: " << bus.fuelAmount() << st::endl;

std::cout << "car fuel amount: " << car.fuelAmount() << std::endl;

std::cout << "truck fuel amount: " << truck.fuelAmount() <<"\n"<< std::endl;

std::cout << "bus capacity: " << bus.capacity() << std::endl;

std::cout << "Car capacity: " << car.capacity() << std::endl;

std::cout << "truck capacity: " << truck.capacity() <<"\n"<< std::endl;

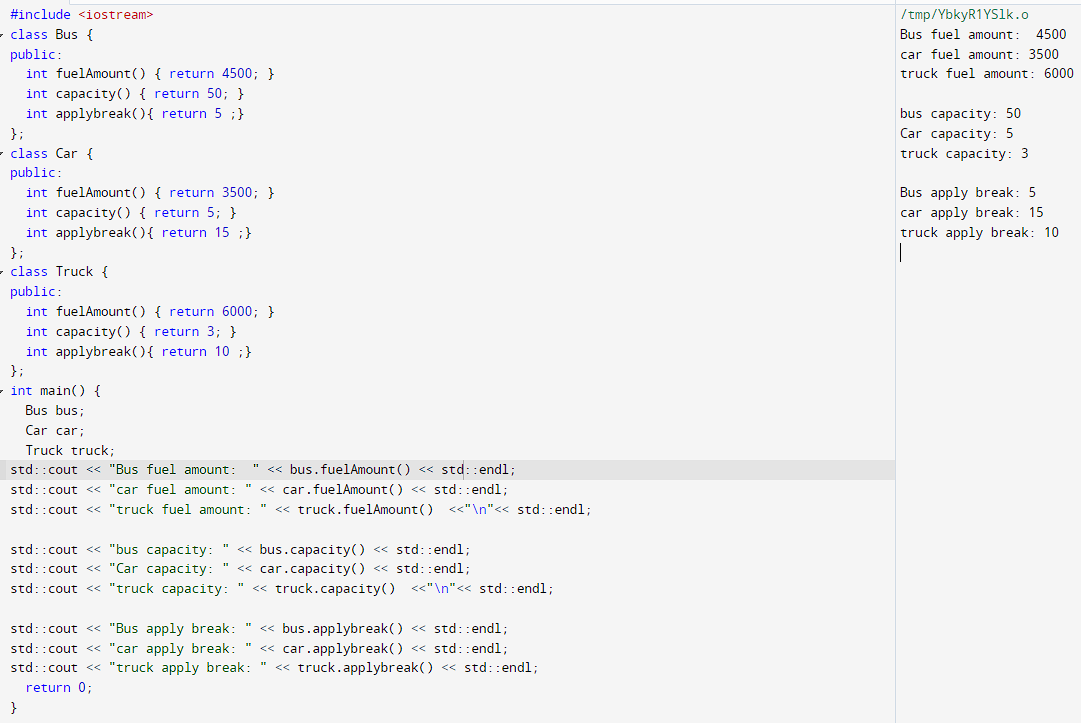
std::cout << "Bus apply break: " << bus.applybreak() << std::endl;

std::cout << "car apply break: " << car.applybreak() << std::endl;

std::cout << "truck apply break: " << truck.applybreak() << std::endl;

return 0;

}



**FN2 task:**

**Multilevel Inheritance Implement a simple Multilevel Inheritance program followings the steps,**

**1. Declare the base class student.**

**2. Declare and define the function get() to get the student details.**

**3. Declare the other class sports.**

**4. Declare and define the function getsm() to read the sports mark.**

**5. Create the class statement derived from students and sports.**

**6. Declare and define the function display() to find out the total and average.**

**7. Declare the derived class object,call the functions get(),getsm() and display().**

#include<iostream>

using namespace std;

class student

{

public:

void get()

{

cout<<"Enter the name of student ";

cin >> name;

cout<< "Enter the roll number of student ";

cin>> rollno;

}

protected:

string name;

int rollno;

};

class sports : public student

{

public:

void getsm()

{

cout<<"Enter the sports mark "<<"\n";

cin>> mark;

}

protected:

float mark;

};

class statement : public sports

{

public:

void display()

{

cout<<"Name "<<name <<endl;

cout<<"Roll number "<<rollno<<endl;

cout<<"Sports mark "<<mark<<endl;

cout<<"Total marks "<<(mark+100)<<endl;

cout<<"Average marks "<< ((mark+100)/2)<<endl;

}

};

int main()

{

statement stud;

stud.get();

stud.getsm();

stud.display();

return 0 ;

}



**AN1 task: Multiple Inheritance Develop an employee payroll management system by implementing the multiple inheritance concepts.**

#include <iostream>

#include <string>

using namespace std;

class Employee {

protected:

string name;

int id;

public:

Employee(string name, int id) {

this->name = name;

this->id = id;}};

class SalariedEmployee : public Employee {

protected:

double salary;

public:

SalariedEmployee(string name, int id, double salary) : Employee(name, id) {

this->salary = salary;}

double getSalary() {

return salary;}};

class HourlyEmployee : public Employee {

protected:

double hourlyRate;

double hoursWorked;

public:

HourlyEmployee(string name, int id, double hourlyRate, double hoursWorked) : Employee(name, id) {

this->hourlyRate = hourlyRate;

this->hoursWorked = hoursWorked;}

double getHourlyRate() {

return hourlyRate;}

double getHoursWorked() {

return hoursWorked;}

double getSalary() {

return hourlyRate \* hoursWorked;}};

class Manager : public SalariedEmployee {

public:

Manager(string name, int id, double salary) : SalariedEmployee(name, id, salary) {}

void printInfo() {

cout << "Name: " << name << ", ID: " << id << ", Salary: " << salary << endl;}

};

class Worker : public HourlyEmployee {

public:

Worker(string name, int id, double hourlyRate, double hoursWorked) : HourlyEmployee(name, id, hourlyRate, hoursWorked) {}

void printInfo() {

cout << "Name: " << name << ", ID: " << id << ", Hourly Rate: " << hourlyRate << ", Hours Worked: " << hoursWorked << endl;

}

};

int main() {

Manager manager1("aaaa", 101, 60000.0);

Manager manager2("bbbb", 102, 70000.0);

Worker worker1("cccc", 201, 25.0, 80.0);

Worker worker2("dddd", 202, 30.0, 45.0);

manager1.printInfo();

manager2.printInfo();

worker1.printInfo();

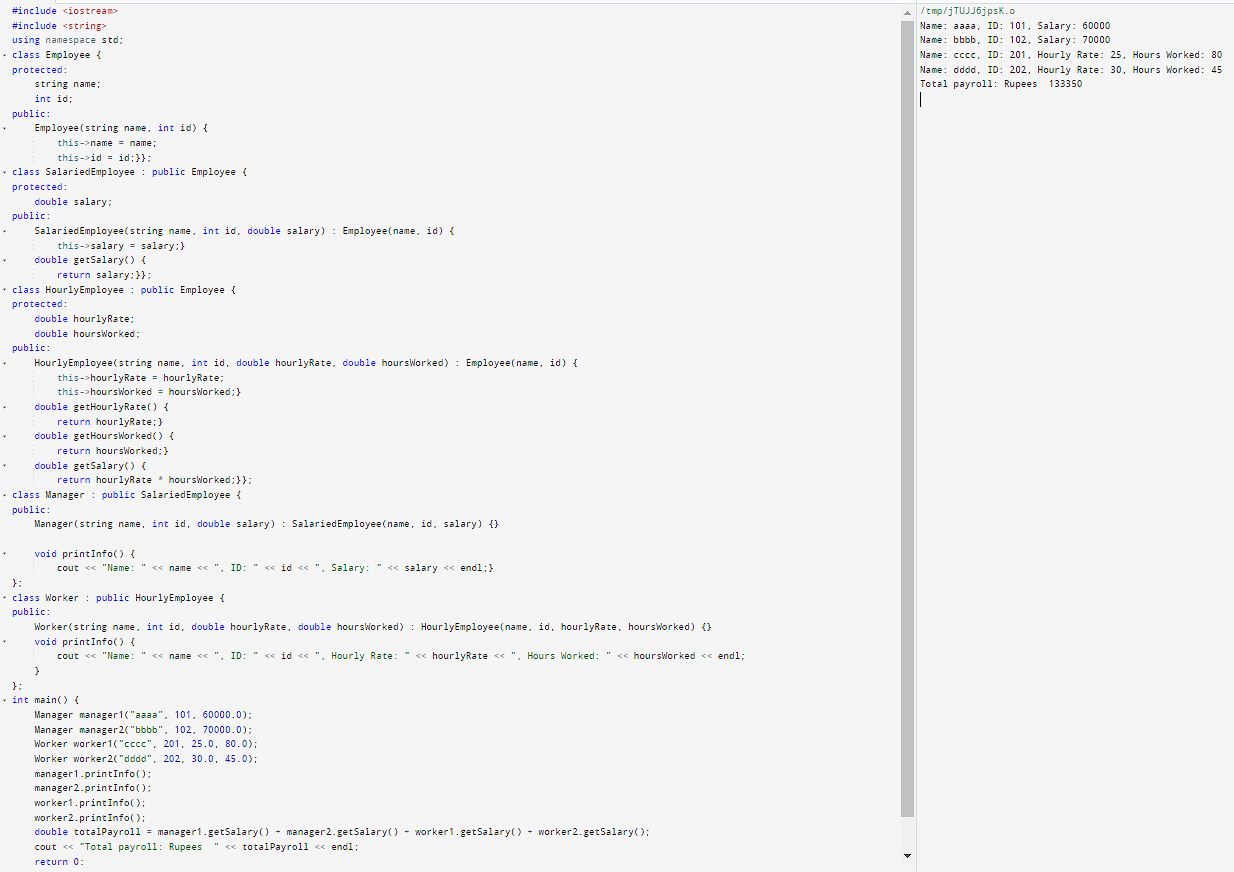
worker2.printInfo();

double totalPayroll = manager1.getSalary() + manager2.getSalary() + worker1.getSalary() + worker2.getSalary();

cout << "Total payroll: Rupees " << totalPayroll << endl;

return 0;

}



**Assessment 1:**

**Online Bus ticket reservation system using multiple inheritance :**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Person {

protected:

string name;

int age;

public:

Person(string name, int age) : name(name), age(age) {}

void display() {

cout << "\nName: " << name << endl;

cout << "Age: " << age << endl;}};

class Reservation {

protected:

string busName;

int busNumber;

int totalSeats;

vector<string> seatsBooked;

public:

Reservation(string busName, int busNumber, int totalSeats) :

busName(busName), busNumber(busNumber), totalSeats(totalSeats) {}

void display() {

cout << "Bus Name: " << busName << endl;

cout << "Bus Number: " << busNumber << endl;

cout << "Total Seats: " << totalSeats << endl;

cout << "Seats Booked: \n";

for (string seat : seatsBooked) {

cout << seat << " "; }

cout << endl; }

bool bookSeat(string seatNumber) {

if (seatsBooked.size() < totalSeats) {

seatsBooked.push\_back(seatNumber);

return true;

}

return false; }};

class Passenger : public Person, public Reservation {

public:

Passenger(string name, int age, string busName, int busNumber, int totalSeats) :

Person(name, age), Reservation(busName, busNumber, totalSeats) {}

void display() {

Person::display();

Reservation::display();

}

bool bookSeat(string seatNumber) {

return Reservation::bookSeat(seatNumber);}};

int main() {

Passenger p1("\n aaaa", 20, "TR BUS", 001, 5);

Passenger p2("\n BBBB", 25, "ROHIT TRAVELS", 201, 10);

p1.display();

p2.display();

p1.bookSeat("1A SLEEPER");

p1.bookSeat("2B AC ");

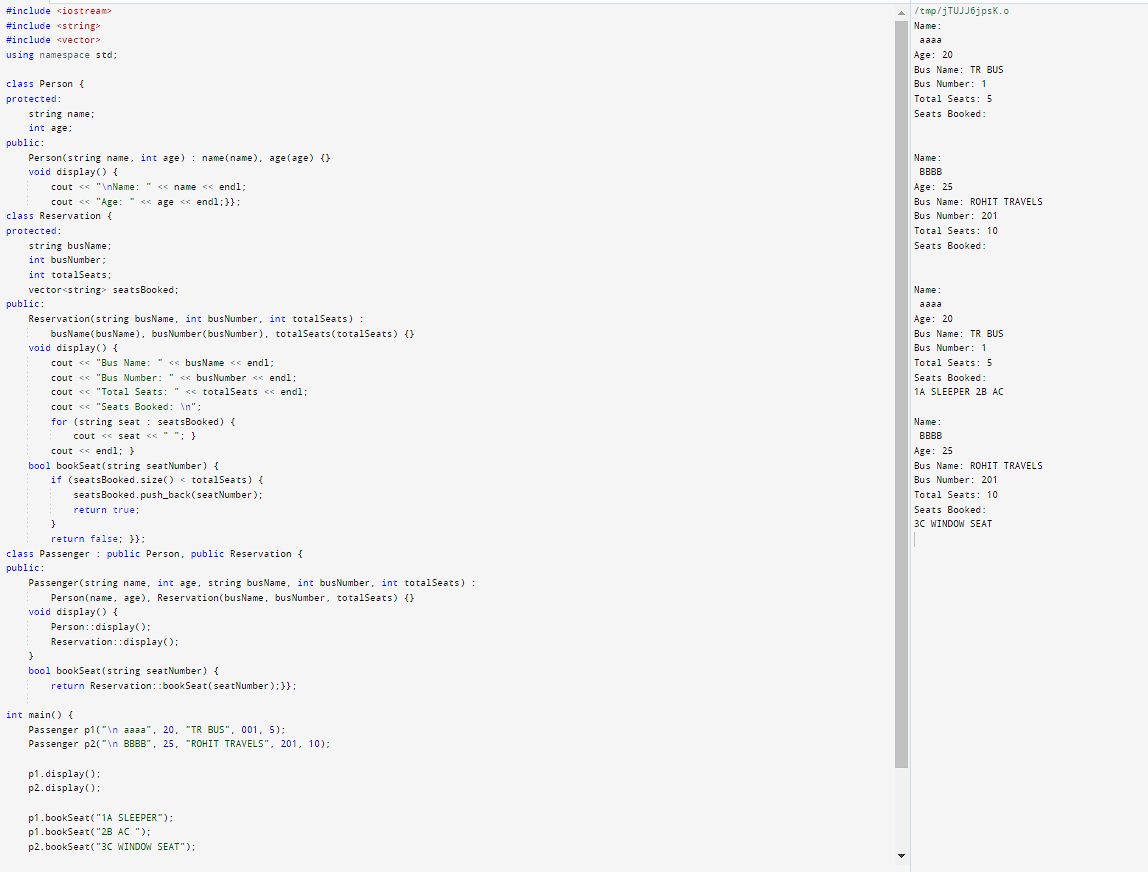
p2.bookSeat("3C WINDOW SEAT");

p1.display();

p2.display();

return 0;

}



**DAY 11:**

**FN1 task:**

**Create a C++ program to overload unary operator ++**

#include <iostream>

class Counter {

public:

Counter() : count(2) {}

Counter& operator++() {

++count;

return \*this;

}

void display() const {

std::cout << "Count: " << count << std::endl;

}

private:

int count;

};

int main() {

Counter c1;

c1.display();

++c1;

c1.display();

++c1;

++c1;

c1.display();

return 0;

}



**FN2 task:**

**Create a C++ program to overload binary operator +**

#include <iostream>

class MyNumber {

public:

MyNumber(int value) : value(value) {}

MyNumber operator+(const MyNumber& other) const {

return MyNumber(value + other.value);

}

void printValue() const {

std::cout << "Value: " << value << std::endl;

}

private:

int value;

};

int main() {

MyNumber n1(3), n2(3);

MyNumber n3 = n1 + n2;

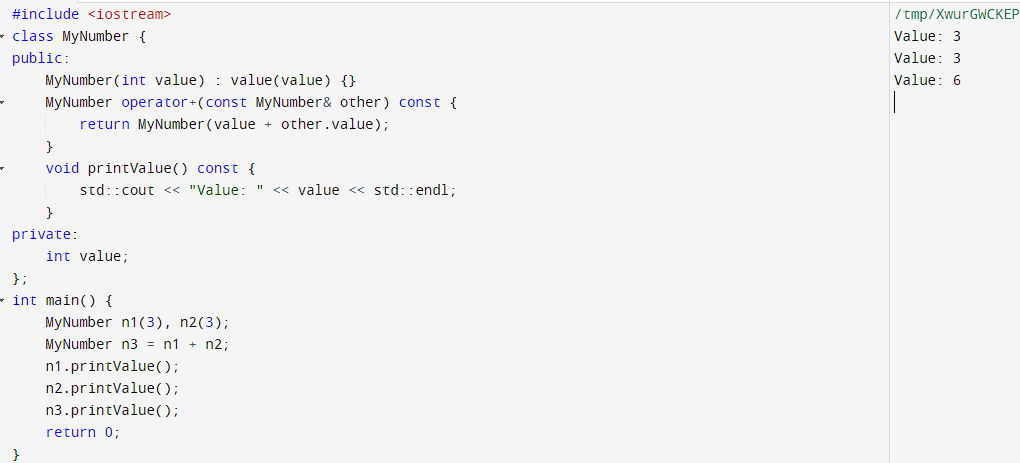
n1.printValue();

n2.printValue();

n3.printValue();

return 0;

}



**2. C++ program for overriding**

#include <iostream>

class Shape {

public:

virtual void draw() const

{

std::cout << "Drawing a shape" << std::endl;

}

};

class Circle : public Shape {

public:

void draw() const override

{

std::cout << "Drawing a circle" << std::endl;

}

};

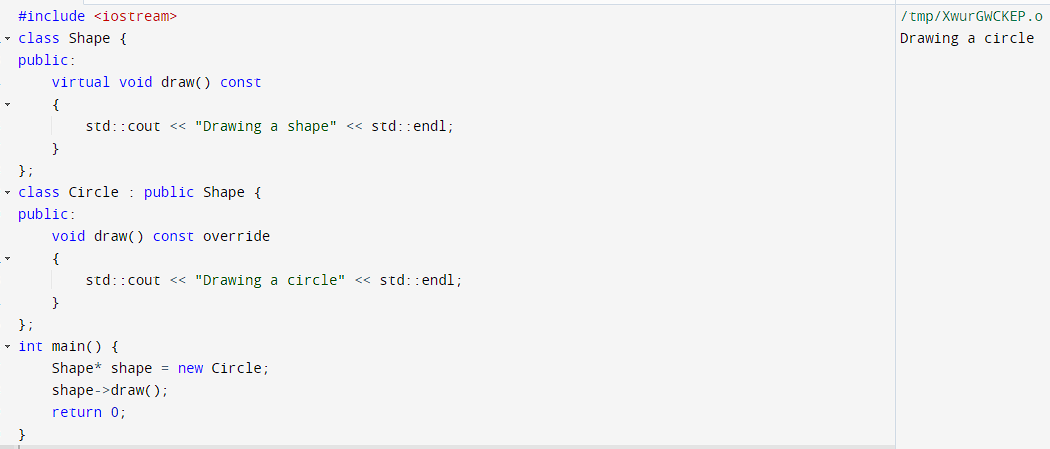
int main() {

Shape\* shape = new Circle;

shape->draw();

return 0;

}



**AN1 task:**

**Write a program to design a class representing the information regarding digital library (books, tape: book & tape should be separate classes having the base class as media). The class should have the functionality for adding new item, issuing, deposit etc. the program should use the runtime polymorphism.**

#include <iostream>

#include <string>

#include <vector>

class Media {

public:

Media(std::string title, std::string author, int year) :

m\_title(title), m\_author(author), m\_year(year) {}

virtual void print() const = 0;

std::string getTitle() const {

return m\_title;

}

protected:

std::string m\_title;

std::string m\_author;

int m\_year;};

class Book : public Media {

public:

Book(std::string title, std::string author, int year, std::string publisher) :

Media(title, author, year), m\_publisher(publisher) {}

void print() const override {

std::cout << "Book: " << m\_title << " by " << m\_author

<< ", published by " << m\_publisher << " in " << m\_year << "\n"<< std::endl;

}

private:

std::string m\_publisher;};

class Tape : public Media {

public:

Tape(std::string title, std::string author, int year, int length) :

Media(title, author, year), m\_length(length) {}

void print() const override {

std::cout << "Tape: " << m\_title << " by " << m\_author

<< ", length " << m\_length << " minutes, released in " << m\_year <<"\n"<< std::endl;

}

private:

int m\_length;};

class Library {

public:

void addMedia(Media\* media) {

m\_media.push\_back(media);

}

void issueMedia(std::string title) {

for (Media\* media : m\_media) {

if (media->getTitle() == title) {

std::cout << "Issuing " << title << "\n"<< std::endl;

return;}

}

std::cout << "Media not found: " << title <<"\n" << std::endl;}

void depositMedia(std::string title) {

std::cout << "Depositing " << title <<"\n"<< std::endl;

}

void listMedia() const {

std::cout << "Library contents: \n" << std::endl;

for (const Media\* media : m\_media) {

media->print(); }

}

private:

std::vector<Media\*> m\_media;};

int main() {

Library library;

Book\* book1 = new Book("The Catcher in the Rye", "J.D. Salinger", 1951, "Little, Brown and Company");

Book\* book2 = new Book("To Kill a Mockingbird", "Harper Lee", 1960, "J. B. Lippincott & Co.");

Tape\* tape1 = new Tape("The Godfather", "Francis Ford Coppola", 1972, 175);

Tape\* tape2 = new Tape("The Shawshank Redemption", "Frank Darabont", 1994, 142);

library.addMedia(book1);

library.addMedia(book2);

library.addMedia(tape1);

library.addMedia(tape2);

library.listMedia();

library.issueMedia("To Kill a Mockingbird");

library.issueMedia("The Shawshank Redemption");

library.issueMedia("The Lord of the Rings");

library.depositMedia("To Kill a Mockingbird");

delete book1;

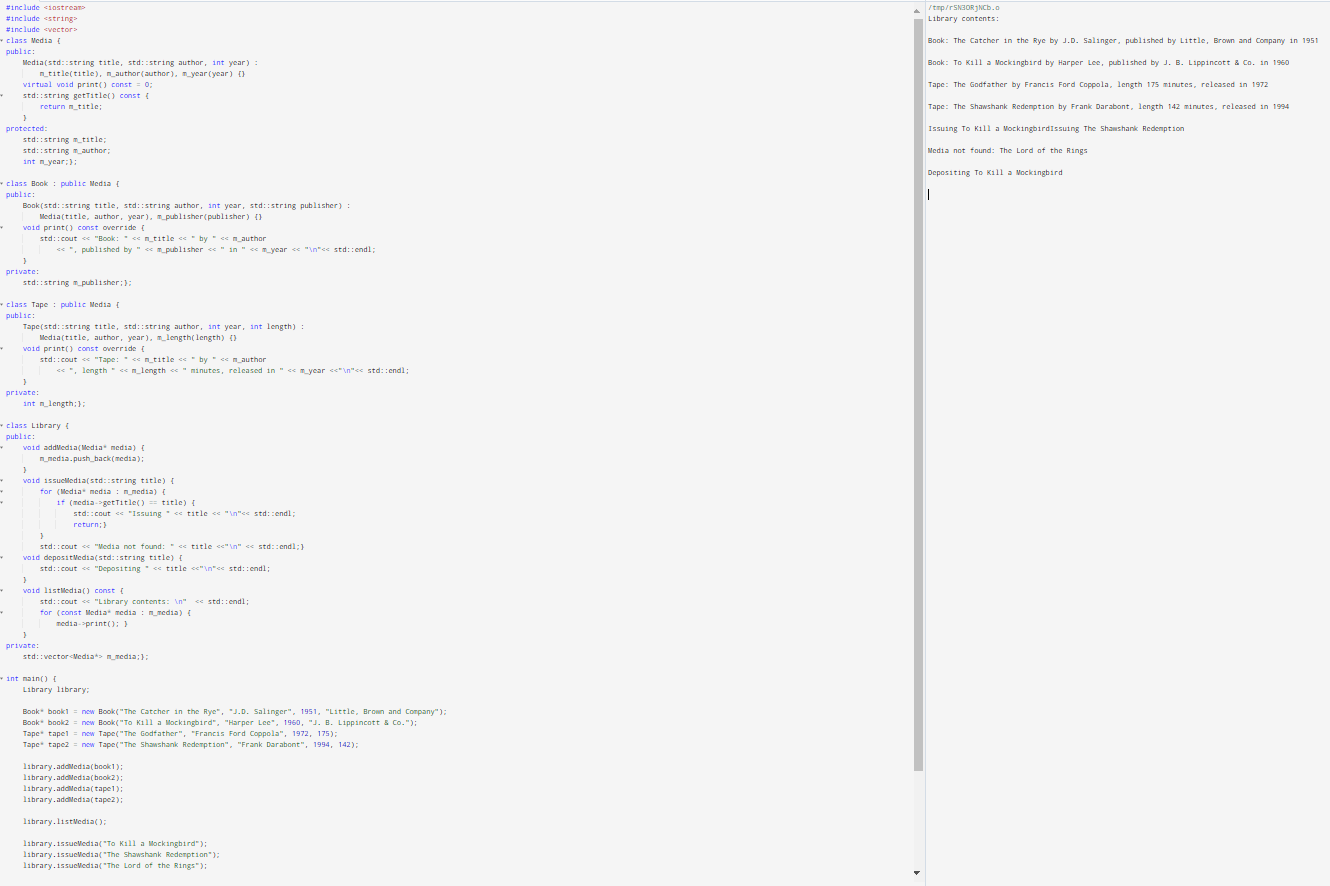
delete book2;

delete tape1;

delete tape2;

return 0;

}



**DAY 12**

**FN1 task: Develop a C++ program to implement the exception handling with**

**multiple catch statements.**

#include <iostream>

using namespace std;

int main() {

int num1, num2;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

try {

if (num2 == 0) {

throw "Division by zero error";

}

else if (num1 == num2) {

throw "Numbers are equal";

}

else {

cout << "Result: " << num1 / num2 << endl;

}

}

catch (const char\* msg) {

cerr << "Error: " << msg << endl;

}

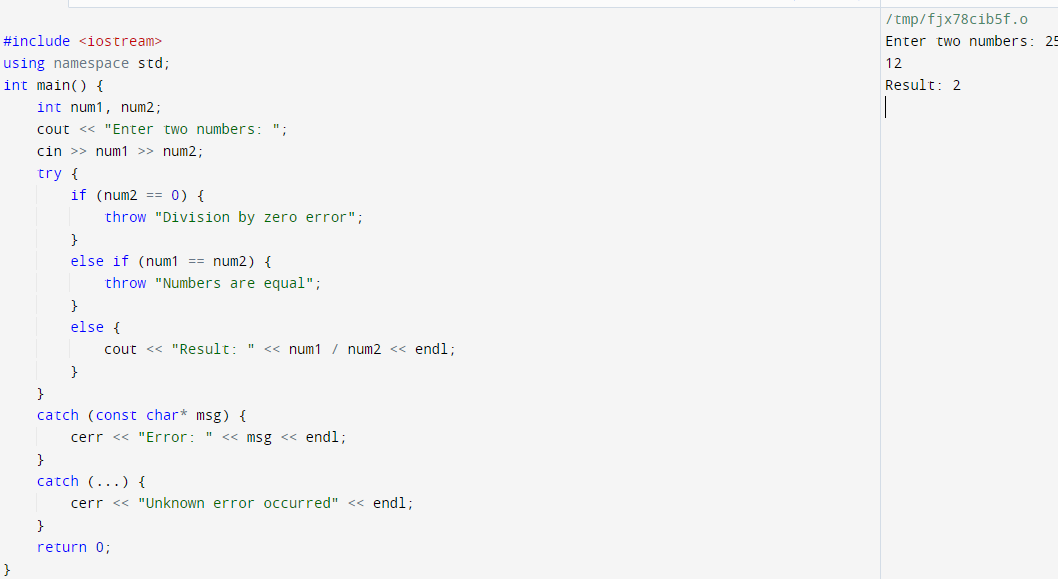
catch (...) {

cerr << "Unknown error occurred" << endl;

}

return 0;

}



**FN2 task: Develop a C++ program to implement the exception handling with re**

**throwing in exception.**

#include <iostream>

using namespace std;

void test() {

try {

throw "Error occurred";

} catch (const char\* msg) {

cout << "Caught exception: " << msg << endl;

throw;

}

}

int main() {

try {

test();

} catch (const char\* msg) {

cout << "Caught exception in main: " << msg << endl;

}

return 0;

}

