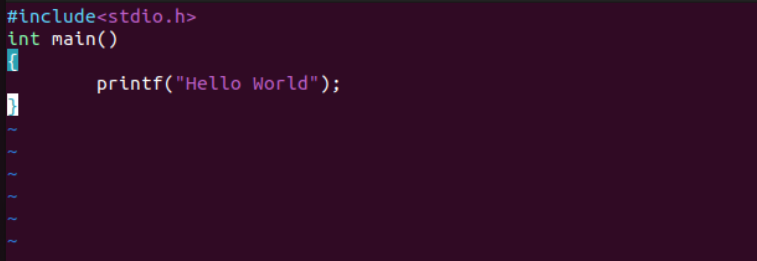
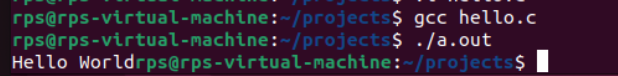
Task:1

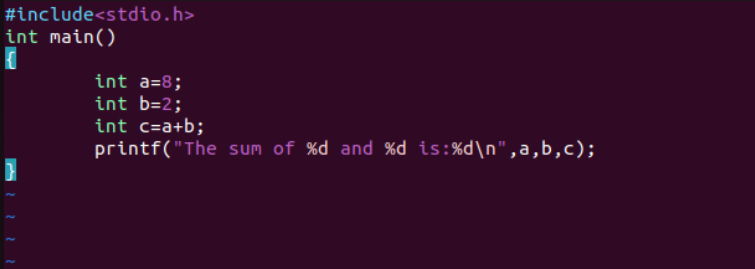
First Program in C

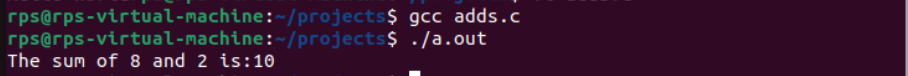




Task:2

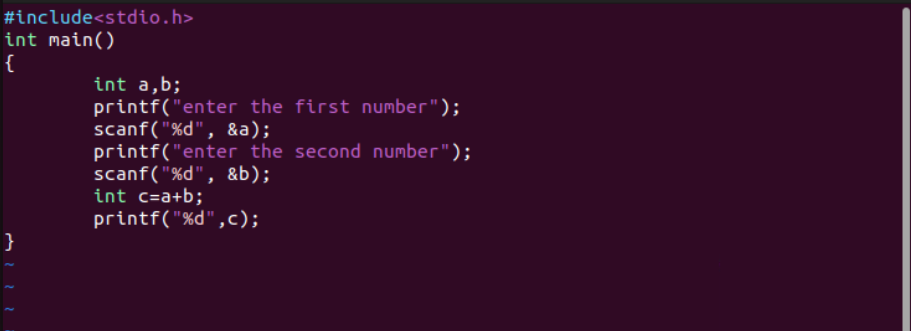
Add two numbers static input

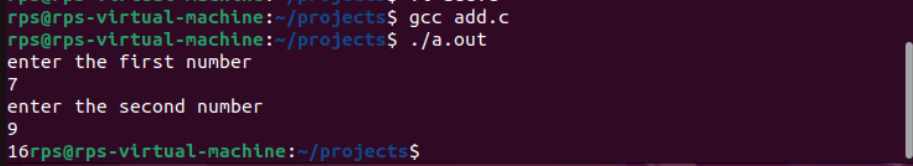




Task:3

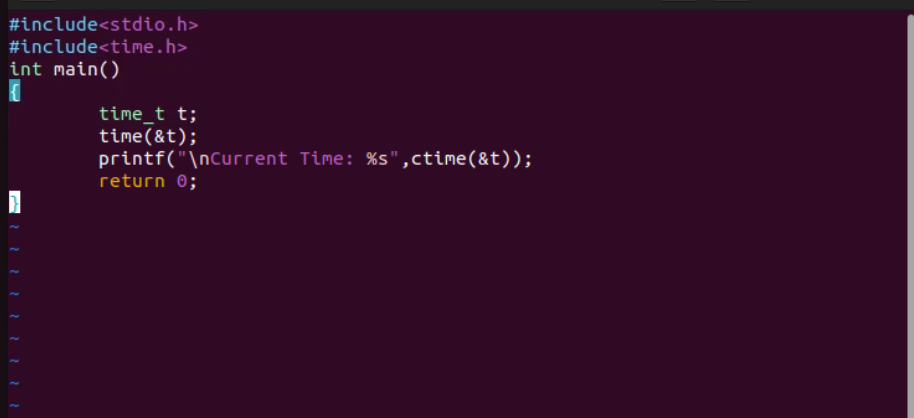
Add two numbers dynamic input

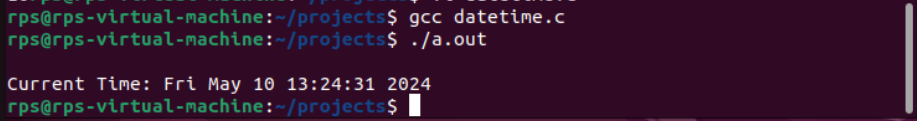




Task:4

Date and Time print on screen

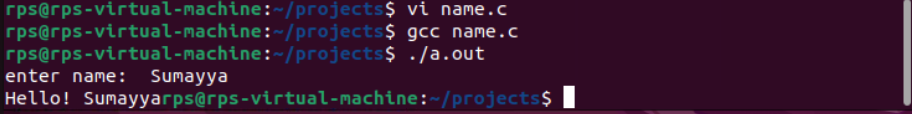




Task:5

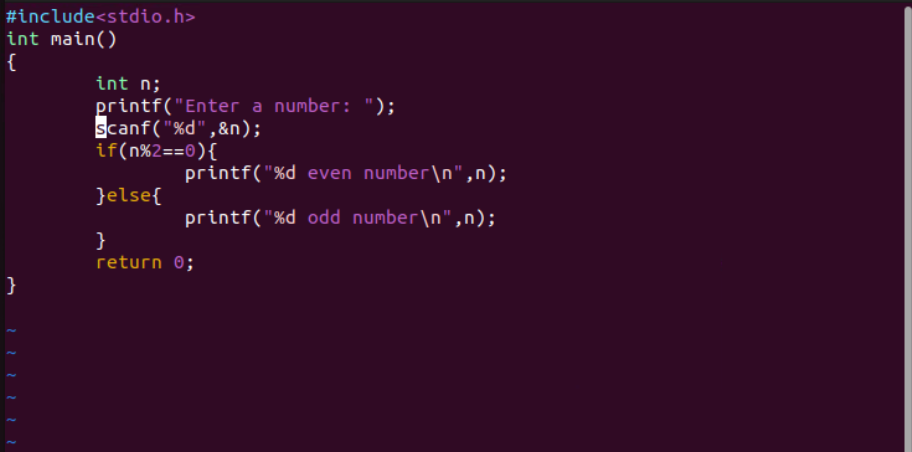
Take input from User to enter their name, and on the screen

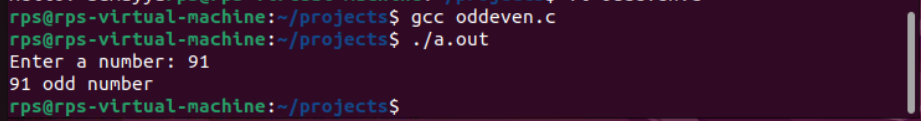




Task:6

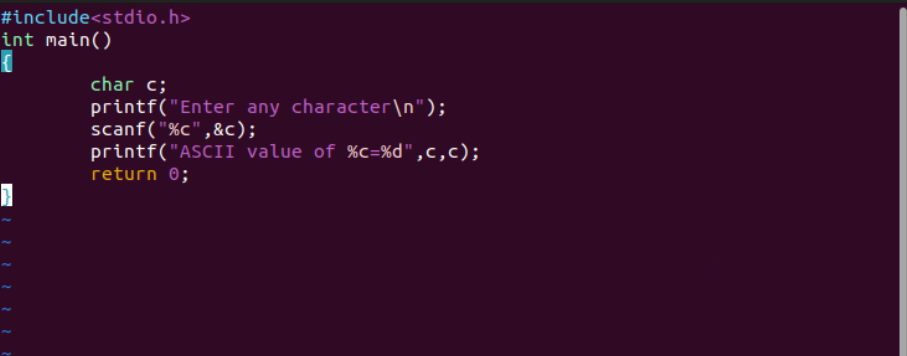
Odd or Even number

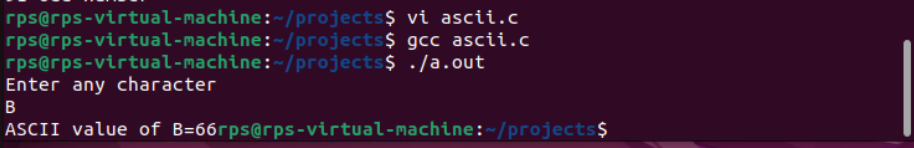




Task:7

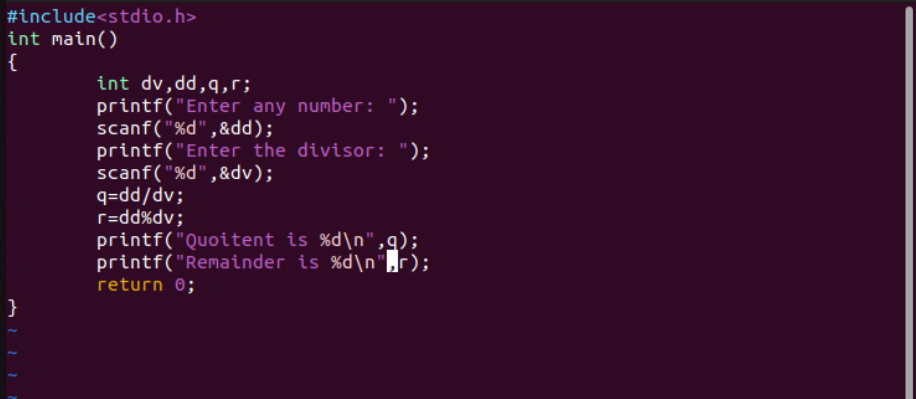
ASCII Value

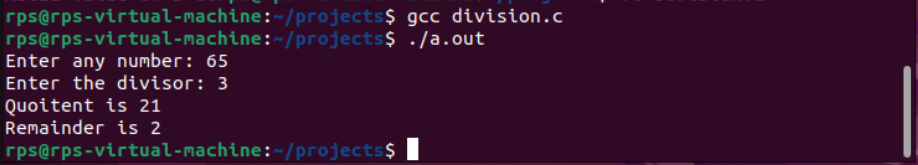




Task:8

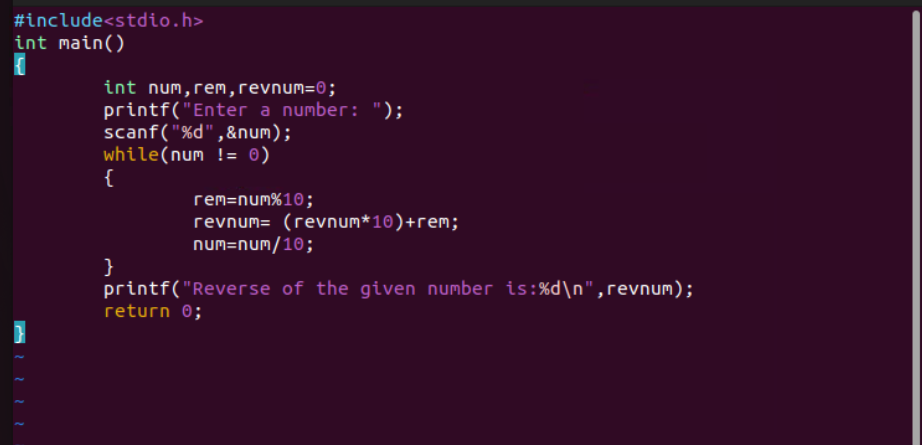
Quotient & Remainder

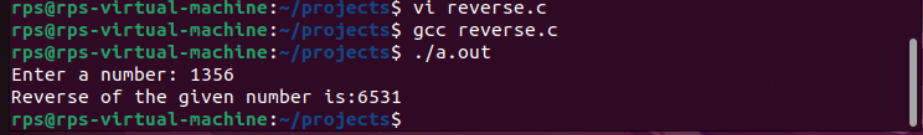




Task:9

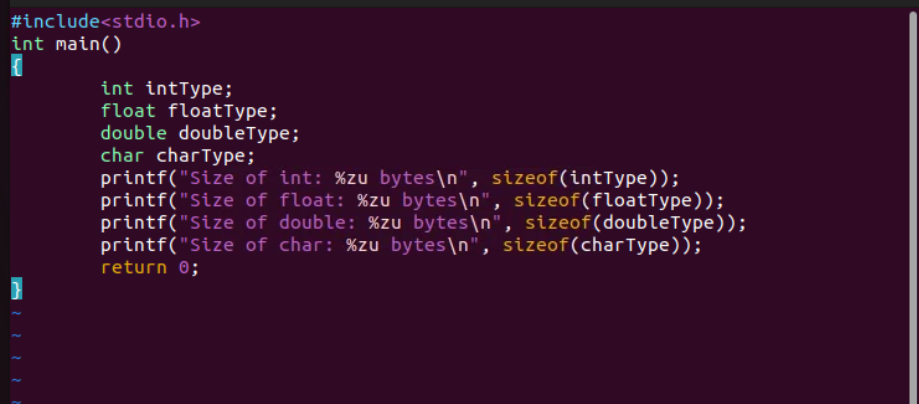
Reverse of a Number

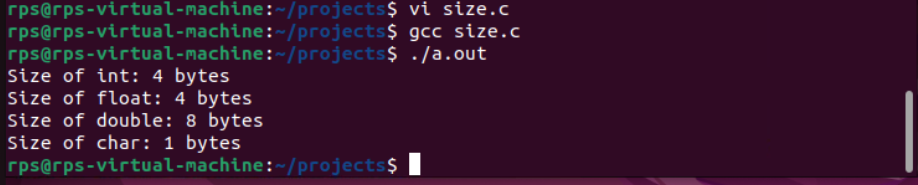




Task:10

To find the size of variables

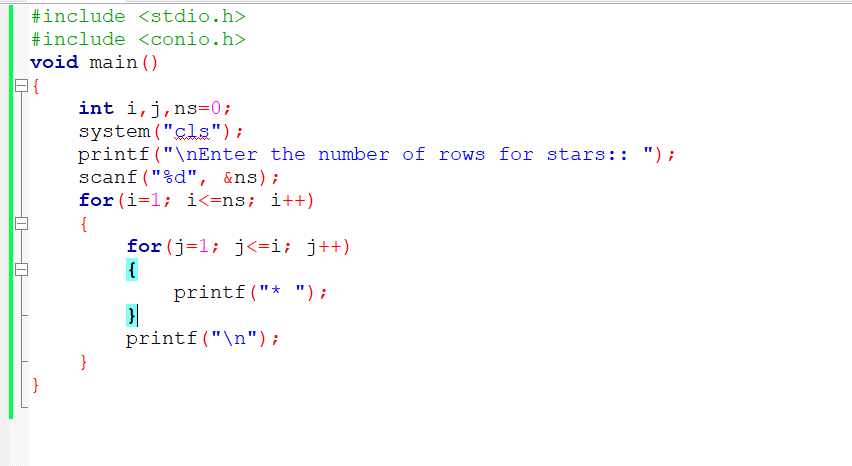




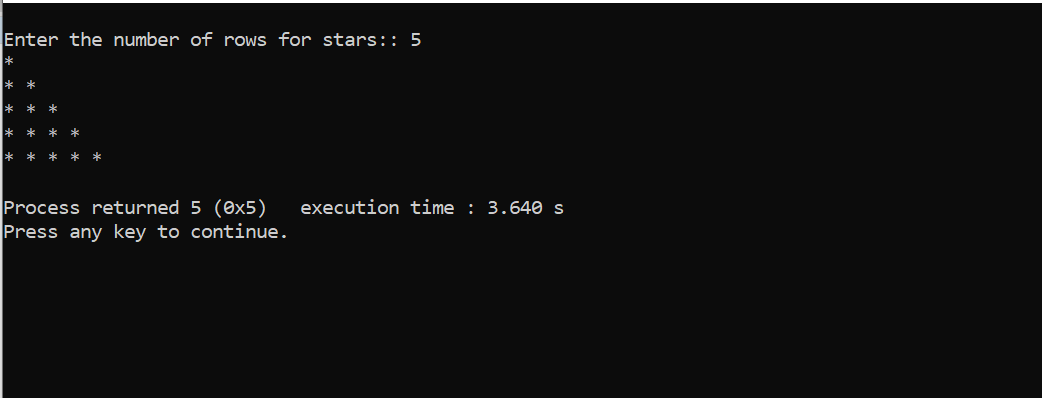
Day-2

Task:1

Printing Stars

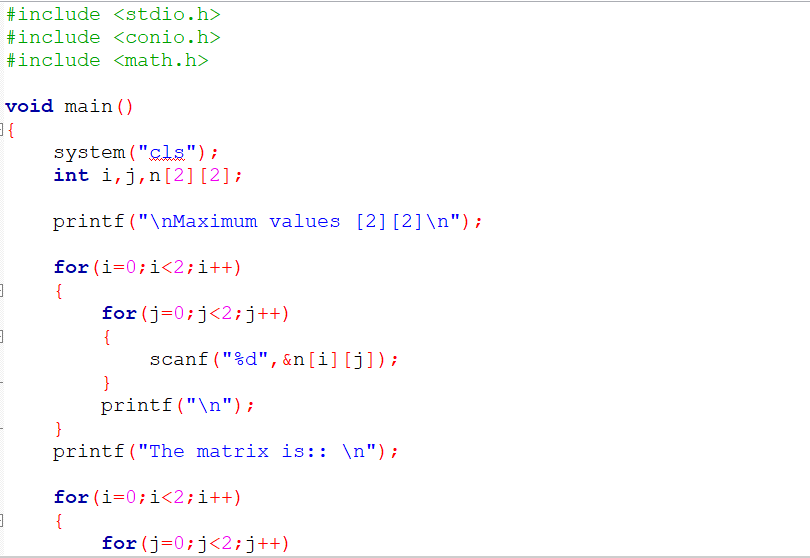


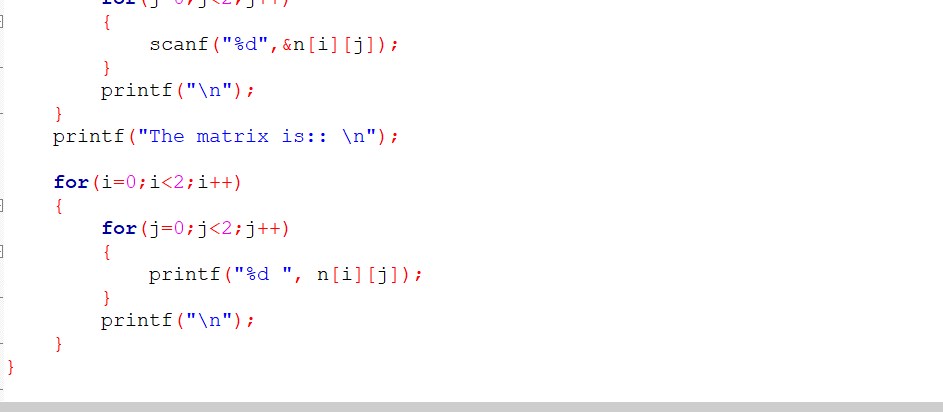
Output:



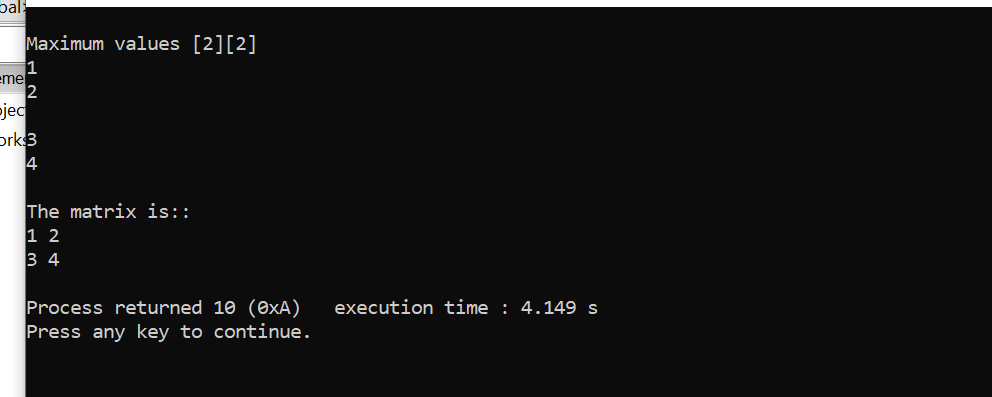
Task:2

Matrix



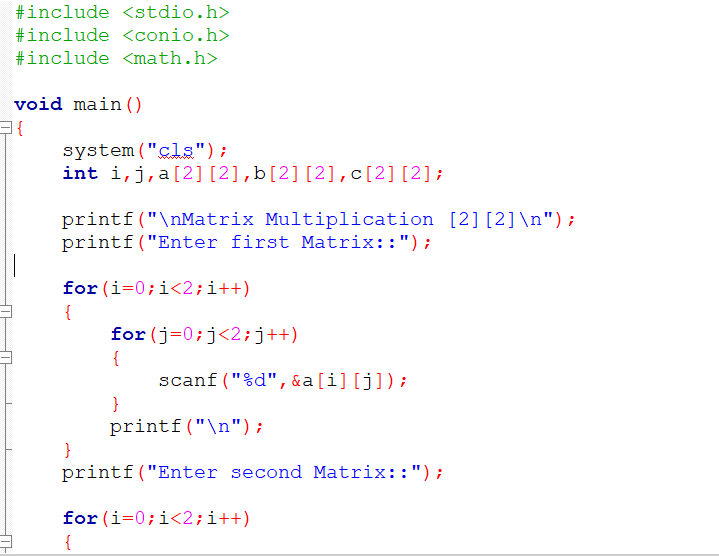


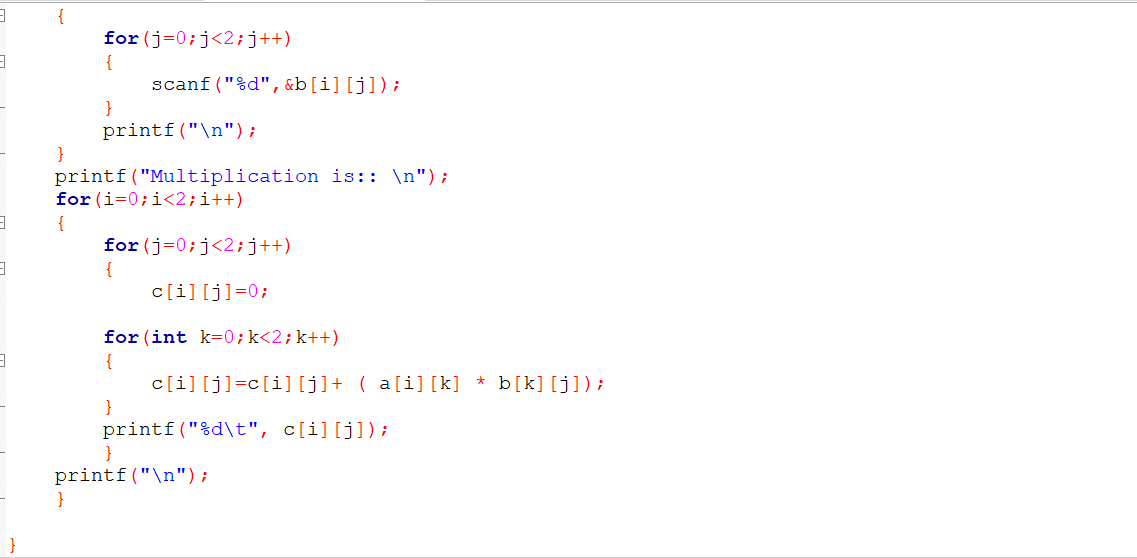
Output:



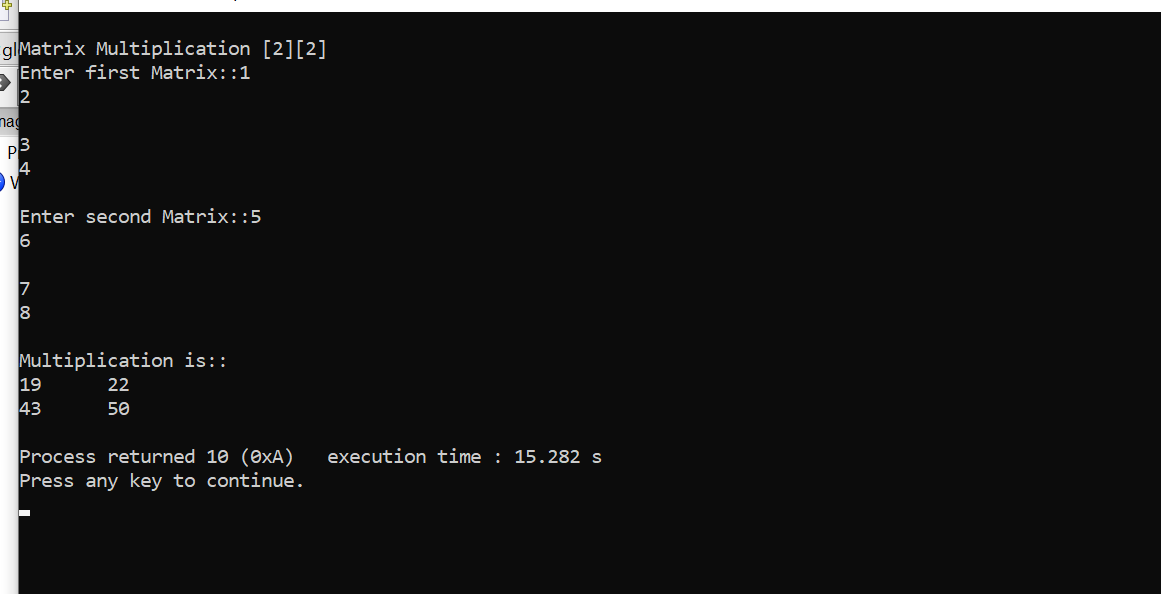
Task:3

Matrix Multiplication



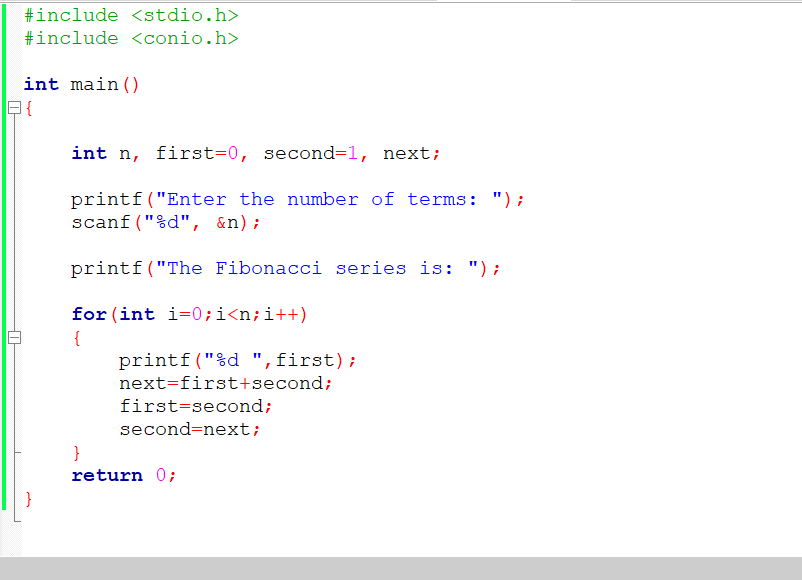


Output:

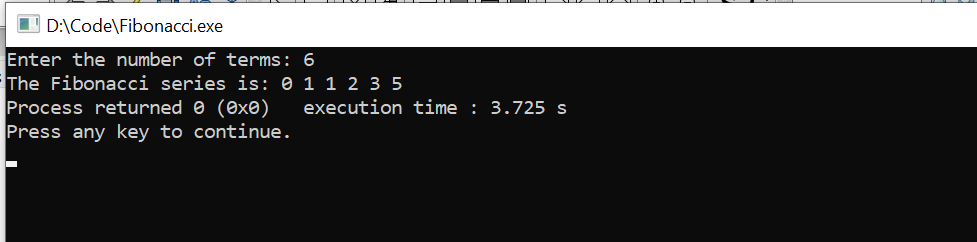


Task: 4

Fibonacci Series

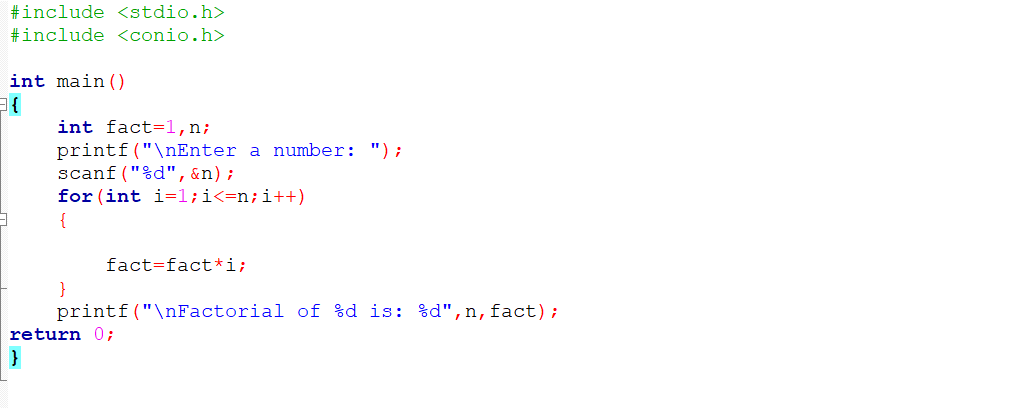


Output:

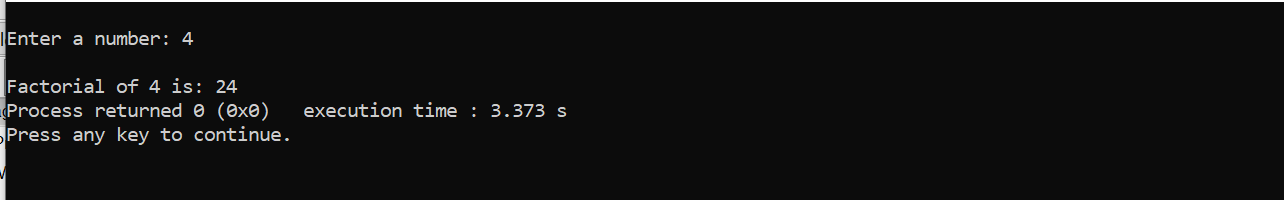


Task: 5

Factorial

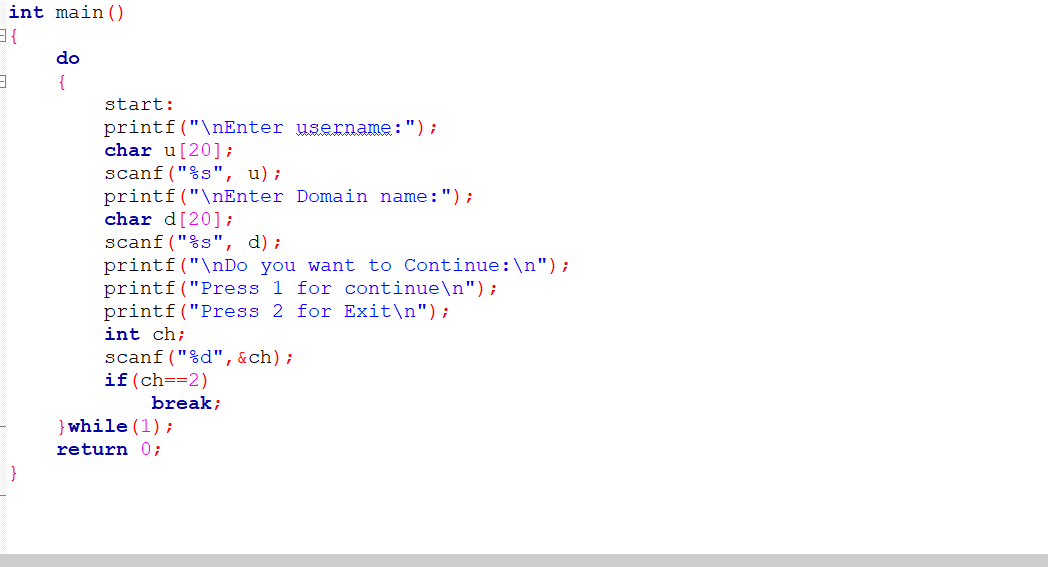


Output:

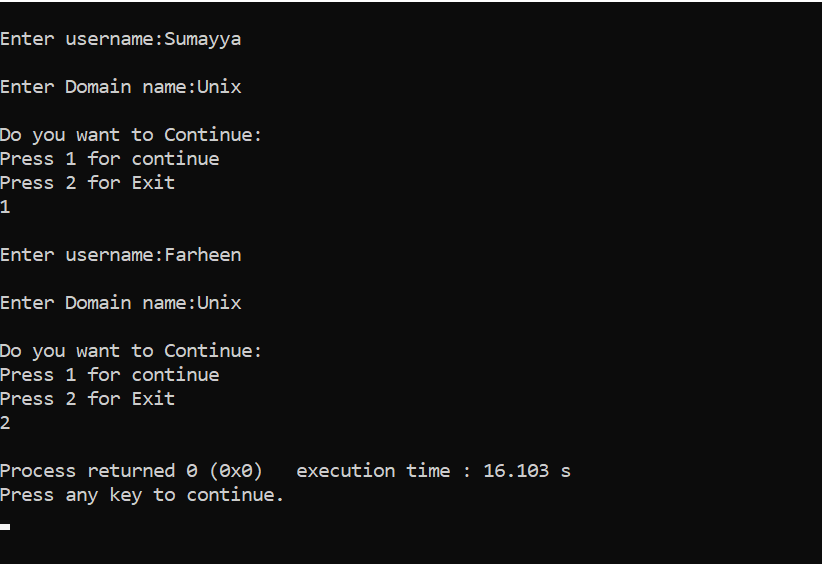


Task:6

Do While Loop

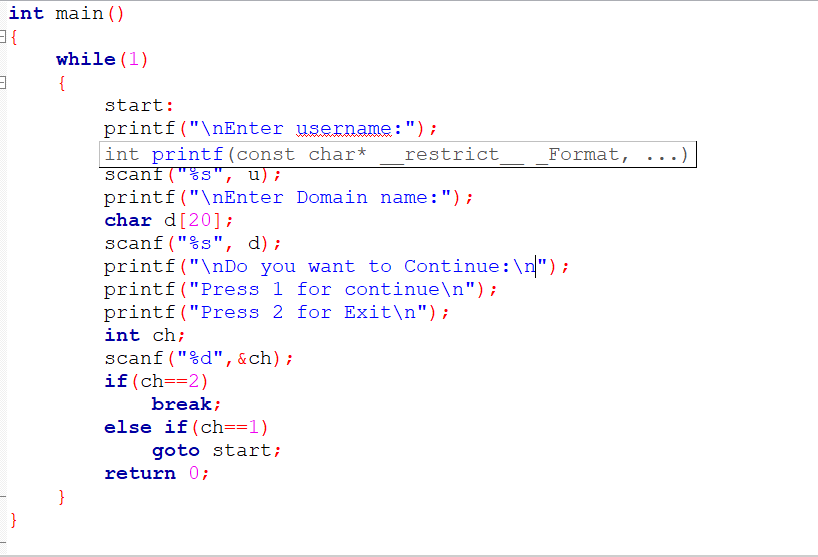


Output:

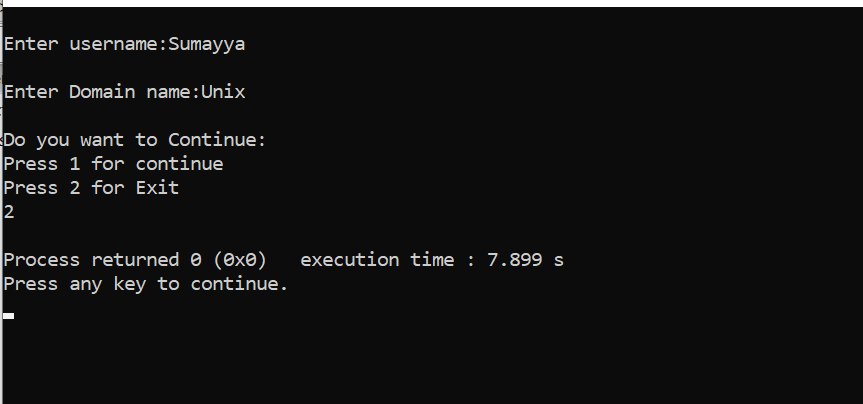


Task:7

While Loop



Output:



Task: 8

a[20] , take the input from user

#include <stdio.h>

void main()

{

int a[20];

printf("Enter 20 elements:\n");

for(int i=0;i<20;i++)

{

printf("Enter element %d: ",i+1);

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

}

printf("\nEntered elements:\n");

for(int i=0; i<20; i++)

{

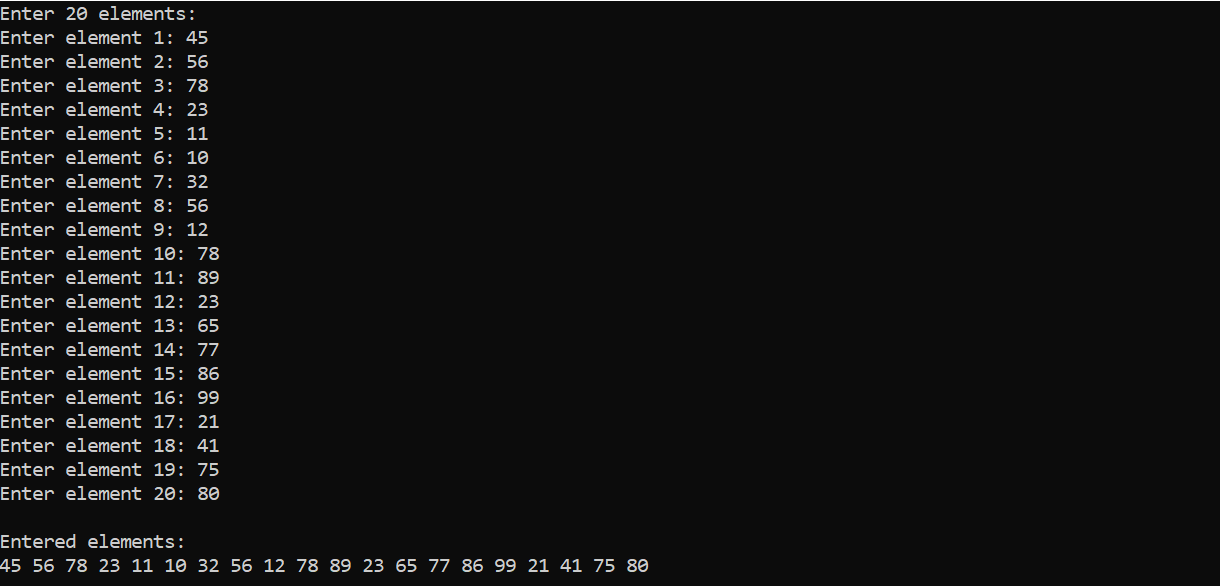
printf("%d ",a[i]);

}

printf("\n");

}

Output:



Task:9

print all the elements pf a[20] with their Index values

#include <stdio.h>

void main()

{

int a[20];

printf("Enter 20 elements:\n");

for(int i=0;i<20;i++)

{

printf("Enter element %d: ",i+1);

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

}

printf("\nEntered elements:\n");

for(int i=0; i<20; i++)

{

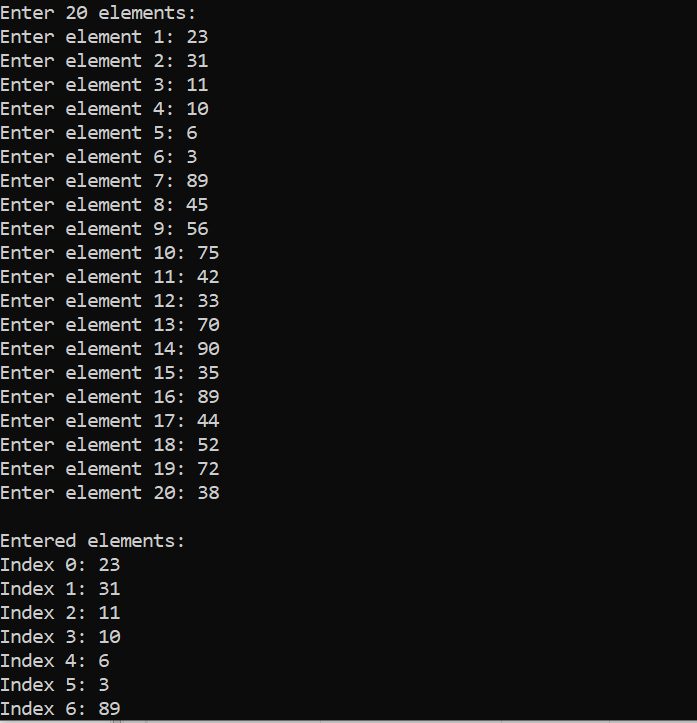
printf("Index %d: %d\n",i,a[i]);

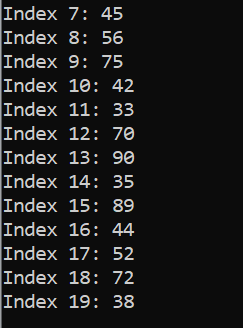
}

printf("\n");

}

Output:





Task: 10

Delete a particular elemenet from an array, a[15]

#include <stdio.h>

void main()

{

int a[20],j;

printf("Enter 20 elements:\n");

for(int i=0;i<20;i++)

{

printf("Enter element %d: ",i+1);

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

}

printf("\nEnter the index value that is to be deleted: ");

scanf("%d",&j);

a[j]=0;

printf("\nThe elements of the array after deletion: \n");

for(int i=0;i<20;i++)

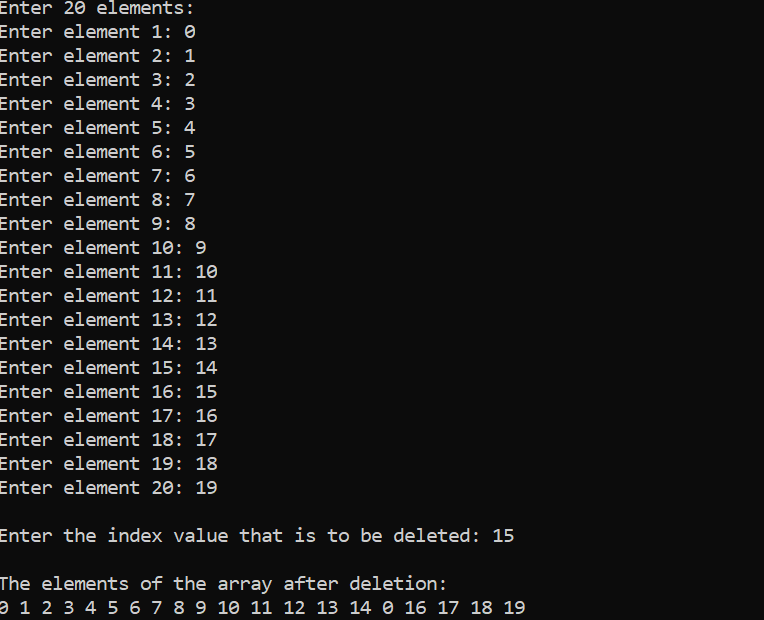
{

printf("%d ",a[i]);

}

}

Output:



Task: 11

Find if there s any duplicates in a[20]

#include <stdio.h>

void main()

{

int a[20];

printf("Enter 20 elements:\n");

for(int i=0;i<20;i++)

{

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

}

printf("\nPrint index values of the duplicate elements: \n");

for(int i=0;i<20;i++)

{

for(int j=i+1;j<20;j++)

{

if(a[i]==a[j])

{

printf("%d and %d\n",i,j);

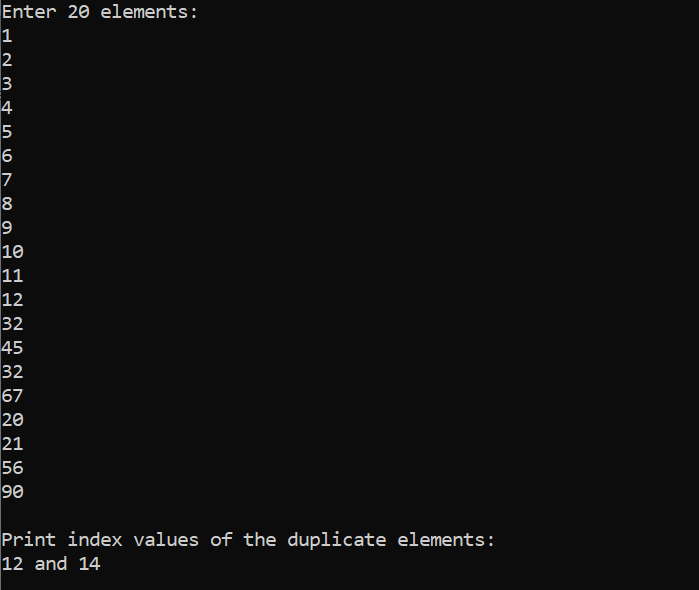
}

}

}

}

Output:



Task: 12

search [10] from a[20]

#include <stdio.h>

void main()

{

int a[20],n,flag=0;

printf("Enter 20 elements:\n");

for(int i=0;i<20;i++)

{

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

}

printf("\nEnter the element that is to be searched: \n");

scanf("%d",n);

for(int i=0;i<20;i++)

{

if(a[i]==n)

{

printf("\nThe given number %d is found at the index value: %d",i);

flag++;

}

}

if(flag==0)

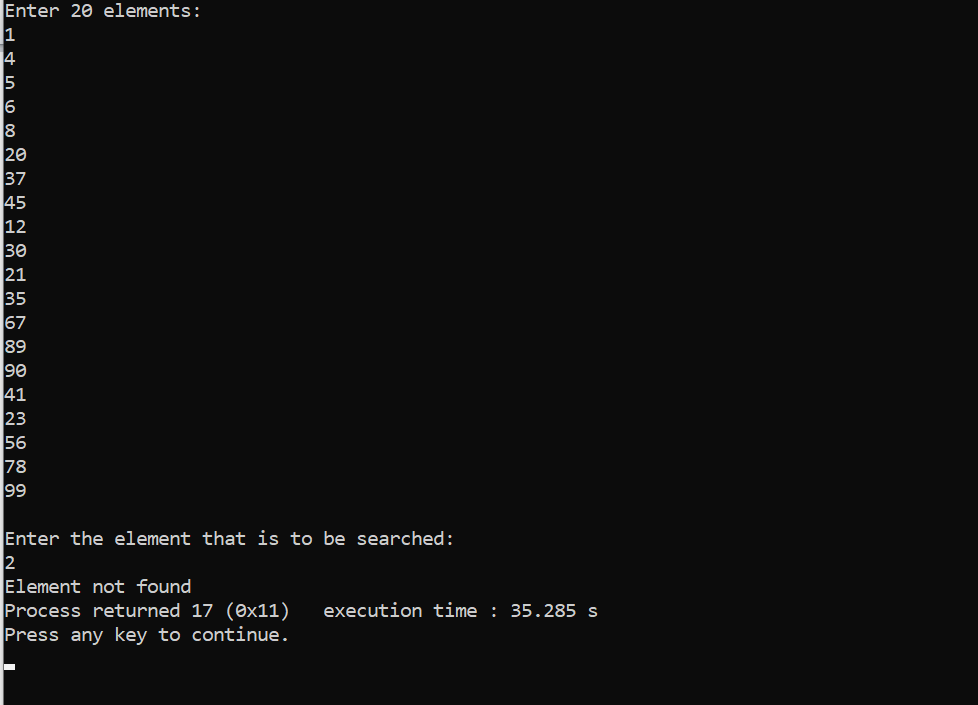
{

printf("Element not found");

}

}

Output:



Pointers:

#include<stdio.h>

void main()

{

int x,y;

int \*ptr;

x=10;

ptr = &x;

y= \*ptr;

printf(" %d (x)is stored in : %u \n",x, &x);

printf(" %d (&x)is stored in : %u \n",&x, &x);

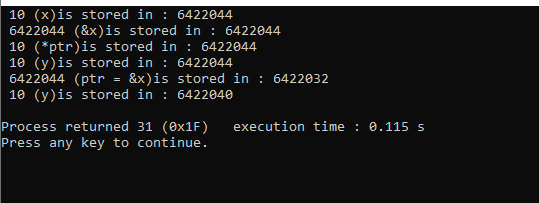
printf(" %d (\*ptr)is stored in : %u \n",\*ptr, ptr);

printf(" %d (y)is stored in : %u \n",y, &\*ptr);

printf(" %u (ptr = &x)is stored in : %u \n",ptr, &ptr);

printf(" %d (y)is stored in : %u \n",y, &y);

}



Size:

#include <stdio.h>

#include <conio.h>

void main()

{

printf ("No. of Bytes occupied by int is %d \n", sizeof(int));

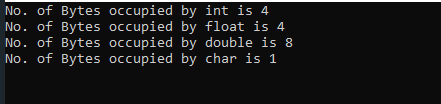
printf ("No. of Bytes occupied by float is %d \n", sizeof(float));

printf ("No. of Bytes occupied by double is %d \n", sizeof(double));

printf ("No. of Bytes occupied by char is %d \n", sizeof(char));

getch();

}



Swap:

#include <stdio.h>

void main()

{

int a,b;

a=5, b=20;

printf ("\n Before Swapping : \n a = %d , b = %d ", a,b);

swap (a,b);

printf ("\n Swap Fun: (call by value) \n a = %d , b = %d ",a,b);

swap1 (&a, &b);

printf ("\n Swap1 Fun: (call by Reference) \n a = %d , b = %d ",a,b);

}

void swap (int x, int y)

{

int tmp;

tmp = x;

x=y;

y=tmp;

}

void swap1 (int \*x1, int \*y1)

{

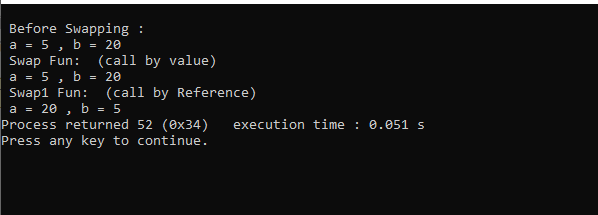
int tmp1;

tmp1 = \*x1;

\*x1=\*y1;

\*y1=tmp1;

}



#include <stdio.h>

float avg (int arr[], int size);

void main ()

{

int x[100], k, n;

printf("\n Enter the array size :\n");

scanf ("%d",&n);

printf("\n Enter the array elements :\n");

for (k=0;k<n;k++)

{

scanf("%d", &x[k]);

}

printf("\n Average is : %f", avg (x,n));

}

float avg (int arr[], int size)

{

int \*p,i,sum=0;

p=arr;

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

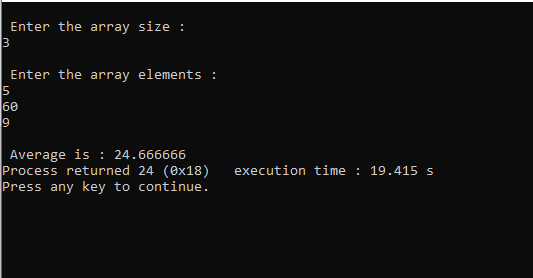
{

sum = sum + \*(p+i);

}

return (float) sum/size;

}



Arrays:

#include <stdio.h>

void main()

{

int arr[5];

int i,j;

printf("Enter the Array Elements::\n");

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

{

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

}

for (j=0;j<5;j++)

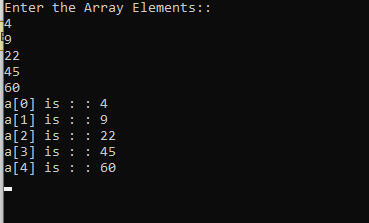
{

printf("a[%d] is : : %d\n",j,arr[j]);

}

getch();

}



Array Deletion:

#include <stdio.h>

void main()

{

int arr[5];

int i,j,n,counter=0,choice;

printf("Enter the Array Elements::\n");

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

{

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

}

for (j=0;j<5;j++)

{

printf("a[%d] is : : %d\n",j,arr[j]);

}

start:

printf("Enter the Array Index you want to delete::\n");

scanf("%d",&n);

arr[n] = 0;

/\*for (i=0;i<=n;i++)

{

if(i==n)

{

arr[i]= 0;

}

}\*/

printf("Array Elements after Deletion::\n");

for (j=0;j<5;j++)

{

printf("a[%d] is : : %d\n",j,arr[j]);

}

printf("\n Do you want to continue deletion: (0/1) : ");

scanf("%d", &choice);

if(choice==1)

{

goto start;

}

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

{

if (arr[i]== 0)

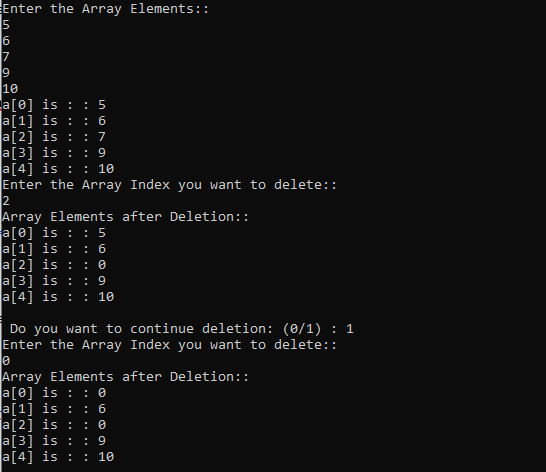
counter = counter +1;

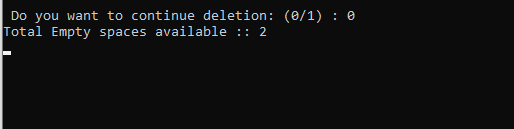
}

printf("Total Empty spaces available :: %d\n", counter);

getch();

}





Airways Reservation System:

#include <stdio.h>

#include <time.h>

#include<ctype.h>

int choice(){

printf("Enter the Month:\n");

int a;

scanf("%d", &a);

printf("============================================\n");

if(a==1)

{

return 0;

}

else if(a==2){

return 1;

}

else if(a==3){

return 2;

}

else if(a==4){

return 3;

}

else if(a==5){

return 4;

}

else if(a==6){

return 5;

}

else if(a==7){

return 6;

}

else if(a==8){

return 7;

}

else if(a==9){

return 8;

}

else if(a==10){

return 9;

}

else if(a==11){

return 10;

}

else if(a==12){

return 11;

}

return -1;

}

int main() {

int year = 2024;

struct tm date = {0};

date.tm\_year = year - 1900;

date.tm\_mon = 0;

date.tm\_mday = 1;

printf("Enter the date to travel: \n");

int da;

scanf("%d", &da);

int a=choice();

if(a==-1){

printf("Enter the correct month: \n");

return 1;

}

date.tm\_mon=a;

mktime(&date);

printf("Sun Mon Tue Wed Thu Fri Sat\n");

for (int i = 0; i < date.tm\_wday; i++) {

printf(" ");

}

int c=0;

while (date.tm\_mon == a) {

printf("%3d ", date.tm\_mday);

if(da==date.tm\_mday){

// printf("Date Found\n");

c=1;

}

if (date.tm\_wday == 6) {

printf("\n");

}

date.tm\_mday++;

mktime(&date);

}

if(c==1){

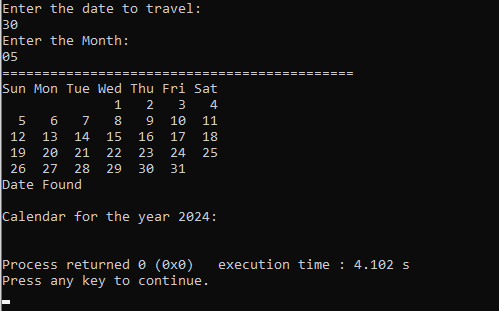
printf("\nDate Found\n");

}

printf("\nCalendar for the year %d:\n\n", year);

return 0;

}



#include<stdio.h>

#include<conio.h>

int dayOfWeek(int d, int m, int y) {

static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};

y -= m < 3;

return (y + y/4 - y/100 + y/400 + t[m-1] + d) % 7;

}

void printCalendar(int month, int year) {

int daysInMonth, i, currentDay;

int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))

days[1] = 29;

printf(" \*\*\*\* Calendar - %d/%d \*\*\*\*\n", month, year);

printf(" Sun Mon Tue Wed Thu Fri Sat\n");

currentDay = dayOfWeek(1, month, year);

for (i = 0; i < currentDay; i++)

printf(" ");

for (i = 1; i <= days[month-1]; i++) {

printf("%5d", i);

if (++currentDay > 6) {

currentDay = 0;

printf("\n");

}

}

if (currentDay != 0)

printf("\n");

}

void main(){

int cho;

system("cls");

printf("\*\*welocme to the airlines\*\*\n");

printf("1.check flight avalilablity\n");

printf("2.book tickets\n");

printf("enter your choice\n");

scanf("%d",&cho);

switch (cho)

{

case 1:

{

int month, year,da,mm;

printf("Enter month and year (MM YYYY) in which you want to travel in the given format:");

scanf("%d %d", &month, &year);

printCalendar(month, year);

printf("choose the date on which you want to travel(dd mm) in the given format");

scanf("%d%d",&da,&mm);

if(da==15||21||03){

FILE \*fp=fopen("D:\\C Programming\\air.txt", "r");

if(fp==NULL){

printf("Error opening File\n");

return ;

}

char a[70];

while(fscanf(fp," %[^\n]",a)!=EOF){

printf("%s\n",a);

}

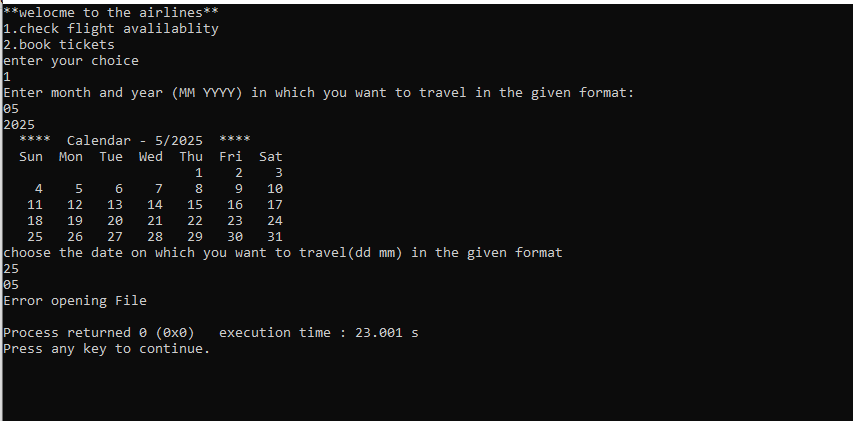
break;

}

}

}

}



Binary Search:

#include <stdio.h>

#include <stdlib.h>

int binarySearch(int array[], int size, int target) {

int left = 0;

int right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (array[mid] == target) {

return mid;

}

if (array[mid] < target) {

left = mid + 1;

}

else {

right = mid - 1;

}

}

return -1;

}

int main() {

int size, target, result;

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

scanf("%d", &size);

int \*array = (int \*)malloc(size \* sizeof(int));

if (array == NULL) {

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

return -1;

}

printf("Enter %d elements in sorted order:\n", size);

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

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

}

printf("Enter the target value to search: ");

scanf("%d", &target);

result = binarySearch(array, size, target);

if (result != -1) {

printf("Element is present at index %d\n", result);

} else {

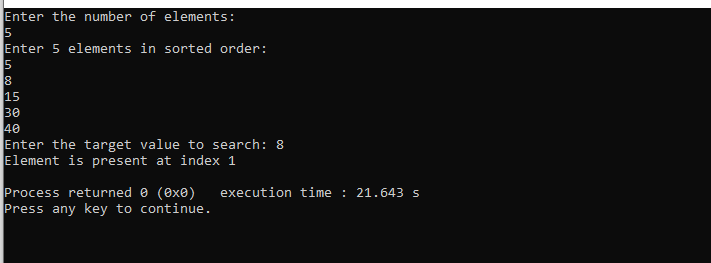
printf("Element is not present in the array\n");

}

free(array);

return 0;

}



Bubble Sort:

#include <stdio.h>

#include <stdlib.h>

void bubbleSort(int arr[], int n) {

int i, j, temp;

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

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

if (arr[j] > arr[j+1]) {

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

void printArray(int arr[], int size) {

int i;

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

printf("%d ", arr[i]);

}

printf("\n");

}

int main() {

int n, i;

int \*arr;

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

scanf("%d", &n);

arr = (int\*)malloc(n \* sizeof(int));

if (arr == NULL) {

printf("Memory allocation failed\n");

return 1;

}

printf("Enter the %d elements: \n", n);

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

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

}

printf("The unsorted array is: \n");

printArray(arr, n);

bubbleSort(arr, n);

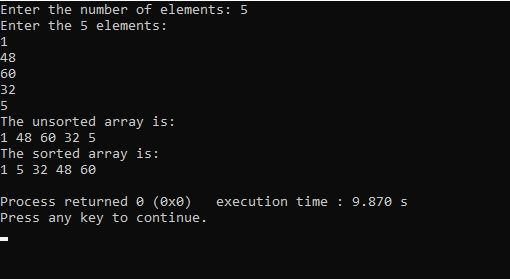
printf("The sorted array is: \n");

printArray(arr, n);

free(arr);

return 0;

}



Bus Reservation:

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

#include <time.h>

typedef struct BinarySearchTree BST;

// if bst is empty then we have to handle the error

struct BinarySearchTree

{

int PassnNo; // busNo0SeatNo.

char name[10];

struct BinarySearchTree \*left;

struct BinarySearchTree \*right;

};

BST \*root = NULL;

int cost(BST \*r); // calculates costs

void status(); // shows bus and seats status

void busLists(); // shows buslist and do booking seat and return customer ID

void DisplaySeat(int bus[33]); // Display the seats of buses

void cancel(int x); //cancel the booking

BST \*reservationInfo(BST \*, int, int \*); // Display Reservation Info

BST \*insert(BST \*\*r, int custID); // inserting a node

int busSeat[32][9] = {0};

void redColor() /// Print the message in redcolor

{

printf("\033[1;31m");

}

void resetColor() /// reset the old color of console

{

printf("\033[0m");

}

BST \*reservationInfo(BST \*r, int s, int \*custIDmatched) // find function

{

if (r == NULL)

return NULL;

BST \*presentnode = r;

while (presentnode)

{

// --------------------

if (presentnode->PassnNo == s)

{

\*custIDmatched = 1;

redColor();

printf("\n-----------------------------------------------------------------");

printf("\n|| NAME: %10s ||", (presentnode->name));

printf("\n|| CUSTOMER ID: %d ||", presentnode->PassnNo);

printf("\n|| BUS NUMBER: %d ||", (presentnode->PassnNo) / 1000);

printf("\n|| SEAT NUMBER: %d ||", (presentnode->PassnNo) % 100);

printf("\n|| TICKET COST: Rs.%d ||", cost(presentnode));

printf("\n-----------------------------------------------------------------");

resetColor();

getch();

return r;

}

else if (presentnode->PassnNo > s)

presentnode = presentnode->left;

else

presentnode = presentnode->right;

}

return NULL;

}

BST \*insert(BST \*\*r, int custId)

{

if (\*r == NULL)

{

\*r = (BST \*)malloc(sizeof(BST));

(\*r)->PassnNo = custId;

if (\*r == NULL)

{

printf("No memory…");

return NULL;

}

else

{

(\*r)->left = (\*r)->right = NULL;

printf("\n ENTER THE PERSON NAME: ");

scanf("%s", &((\*r)->name));

}

}

else

{

if ((\*r)->PassnNo > custId)

{

(\*r)->left = insert(&((\*r)->left), custId);

}

else if ((\*r)->PassnNo < custId)

{

(\*r)->right = insert(&((\*r)->right), custId);

}

}

return \*r;

}

void DisplaySeat(int bus[33])

{

for (int i = 1; i <= 32; i++)

{

redColor();

if (i < 10 && i > 0)

{

printf("0%d .", i);

}

else

{

printf("%d .", i);

}

resetColor();

{

if (bus[i] == 0)

printf("EMPTY ");

else

printf("BOOKED"); // reserv

}

printf(" ");

if (i % 4 == 0)

printf("\n");

}

}

void login()

{

char userName[20] = "user";

char passWord[10] = "team18";

char matchPass[10];

char matchUser[10];

int value;

redColor();

printf("\n\n=========================================================================================\n");

printf("\n\t\t\tWELCOME TO ONLINE BUS RESERVATION");

printf("\n\n=========================================================================================\n\n");

resetColor();

login:

{

printf("\n\nUserName: ");

gets(matchUser);

printf("\nPassWord: ");

gets(matchPass);

}

value = strcmp(passWord, matchPass); /// string compare is function defined in headerfile i.e string.h

if (value != 0)

{

redColor();

printf("\nINVALID DETAILS TRY AGAIN...\n");

resetColor();

goto login;

}

else

{

printf("\nLOGED IN SUCCESFULLY...\n");

}

}

int cost(BST \*r)

{

int cost, buscost;

buscost = (r->PassnNo) / 1000;

switch (buscost % 3)

{

case 1:

return 70;

break;

case 2:

return 55;

break;

case 0:

return 40;

break;

default:

return 0;

break;

}

}

void status()

{

int busNum;

busLists();

busInput:

printf("\n\nENTER YOUR BUS NUMBER : ");

scanf("%d", &busNum);

if (busNum <= 0 || busNum >= 10)

{

redColor();

printf("\n PLEASE ENTER CORRECT BUS NUMBER !!\n");

resetColor();

goto busInput;

}

printf("\n");

DisplaySeat(busSeat[busNum]);

getch();

}

void busLists()

{

redColor();

printf("-----------------------------------------------------------------------------------------");

printf("\nBus.No\tName\t\t\tDestinations \t\tCharges \t\tTime\n");

printf("-----------------------------------------------------------------------------------------");

resetColor();

printf("\n1\tGangaTravels \tDharan to Kavre \tRs.70 \t\t07:00 AM");

printf("\n2\tPhaphara Travels \tKavre To Dharan \tRs.55 \t\t01:30 PM");

printf("\n3\tShiv Ganga Travels \tAllahabad To Gorakhpur\tRs.40 \t\t03:50 PM");

printf("\n4\tSuper Deluxe \tPokhara To Benigha \tRs.70 \t\t01:00 AM");

printf("\n5\tSai Baba Travels \tMaitidevi To Janakpur \tRs.55 \t\t12:05 AM");

printf("\n6\tShine On Travels \tMadhubani to Patna \tRs.40 \t\t09:30 AM");

printf("\n7\tMayur Travels \tPatna To Gaya \tRs.70 \t\t11:00 PM");

printf("\n8\tRajjo Travels \tBegusarai To Patna \tRs.55 \t\t08:15 AM");

printf("\n9\tShree Travels \tGaya To Chhapra \tRs.40 \t\t04:00 PM");

printf("\n");

printf("\n PRESS 'ENTER' KEY TO CONTINUE ");

getch();

}

void cancel(int randomNum)

{

int reservationNo;

int seatNumber;

int choice;

char c;

int seatCancel;

aa:

{

printf("\nENTER YOUR RESERVATION NUMBER : ");

scanf("%d", &reservationNo);

if (reservationNo == randomNum)

{

printf("\nRESERVATION NUMBER IS IT CORRECT ? %d \nENTER (Y/N) : ", reservationNo);

scanf("%s", &c);

if (c == 'y' || c == 'Y')

{

printf("\n\n============================================\n\n");

printf(" ENTER THE BUS NUMBER: ");

scanf("%d", &choice);

printf("\n HOW MANY SEATS DO WANT TO CANCEL : ");

scanf("%d", &seatCancel);

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

{

printf(" \nENTER THE SEAT NUMBER: ");

scanf("%d", &seatNumber);

busSeat[choice][seatNumber] = 0;

}

printf("\n\nYOUR RESERVATION HAS BEEN CANCEL !!\n\n");

printf("\n PRESS 'ENTER' KEY TO CONTINUE \n");

getch();

DisplaySeat(busSeat[choice]);

}

else if (c == 'n' || c == 'N')

{

printf("\nYOUR RESERVATION CANCELATION HAS BEEN DENIED\n");

}

}

else

{

printf("\nNOT FOUND!! ENTER THE CORRECT RESERVATION NUMBER\n");

goto aa;

}

}

}

int main()

{

srand(time(0));

int randomNum = rand();

int num, i, custID, reservationNo;

BST \*root1;

login();

main:

{

do

{

printf("\n\n====================================================================\n\n");

printf("\t\t\t\033[1;31mBUS RESERVATION\033[0m\t\t");

printf("\n\n=====================================================================\n");

printf("\n====================");

redColor();

printf(" MAIN MENU ");

resetColor();

printf("=====================\n\n");

printf(" \033[1;31m[1]\033[0m VIEW BUS LIST \n\n");

printf(" \033[1;31m[2]\033[0m BOOK TICKETS\n\n");

printf(" \033[1;31m[3]\033[0m CANCEL BOOKING\n\n");

printf(" \033[1;31m[4]\033[0m BUSES SEATS INFO\n\n");

printf(" \033[1;31m[5]\033[0m RESERVATION INFO\n\n");

printf(" \033[1;31m[6]\033[0m EXIT\n");

printf("\n=====================================================\n");

printf("\n ENTER YOUR CHOICE: ");

scanf("%d", &num);

switch (num)

{

case 1:

busLists(); // for list of bus

break;

case 2:

busLists(); // for booking the tickets

int CustId, choice, seats;

busChoice:

printf("\n\nCHOOSE YOUR BUS : ");

scanf("%d", &choice);

if (choice <= 0 || choice > 9)

{

redColor();

printf("\nENTER VALID BUS NUMBER !! \n");

resetColor();

getch();

goto busChoice;

}

printf("\n");

DisplaySeat(busSeat[choice]);

busSeatChoice:

printf("\n\nNO. OF SEATS YOU NEED TO BOOK : ");

scanf("%d", &seats);

if (seats <= 0)

{

redColor();

printf("\nENTER VALID SEAT NUMBER!!\n");

resetColor();

goto busSeatChoice;

}

else if (seats > 32)

{

redColor();

printf("\nENTER VALID SEAT NUMBER WE HAVE ONLY 32 SEATS IN A BUS !!\n");

resetColor();

goto busSeatChoice;

}

int seatNumber;

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

{

printf("\n\n==================================================================================\n\n");

seat:

printf(" ENTER THE SEAT NUMBER: ");

scanf("%d", &seatNumber);

if (seatNumber <= 0)

{

redColor();

printf("\n ENTER VALID SEAT NUMBER!!\n\n");

resetColor();

goto seat;

}

else if (seatNumber > 32)

{

redColor();

printf("\n ENTER VALID SEAT NUMBER WE HAVE ONLY 32 SEATS IN A BUS !!\n\n");

resetColor();

goto seat;

}

CustId = choice \* 1000 + seatNumber; // CustumerId

busSeat[choice][seatNumber] = 1;

root = insert(&root, CustId);

redColor();

printf("\n YOUR CUSTOMER ID IS : %d", CustId);

resetColor();

printf("\n\n==================================================================================\n\n");

}

printf("\nYOUR RESERVATION NUMBER IS : ");

redColor();

printf("%d\n", randomNum);

printf("\nPLEASE NOTE DOWN YOUR RESERVATION NUMBER FOR CANCEL BOOKING TICKETS!!\n");

resetColor();

printf("PRESS 'ENTER' KEY TO CONTINUE ");

getch();

break;

case 3:

cancel(randomNum);

break;

case 4:

status(randomNum);

break;

case 5:

takingReservationNo:

printf("\n ENTER YOUR RESERVATION NUMBER :");

scanf("%d", &reservationNo);

if (randomNum == reservationNo)

{

cust:

printf("\n ENTER YOUR CUSTOMER ID :");

scanf("%d", &custID);

int custIDmatched = 0;

root1 = reservationInfo(root, custID, &custIDmatched);

if (custIDmatched == 0)

{

redColor();

printf("\n ENTER CORRECT CUSTOMER ID!!\n");

resetColor();

goto cust;

}

}

else

{

redColor();

printf("\n INVALID RESERVATION NUMBER PLEASE ENTER CORRECT RESERVATION NUMBER !!\n");

resetColor();

goto takingReservationNo;

}

break;

default:

redColor();

printf("\n\n INVALID INPUT CHOOSE CORRECT OPTION\n");

resetColor();

break;

}

} while (num != 6);

printf("\n\n=====================================================================\n\n");

printf("THANK YOU FOR USING THIS BUS RESERVATION SYSTEM");

printf("\n\nPRESS ANY KEY TO EXIT THE END PROGRAM !! \n");

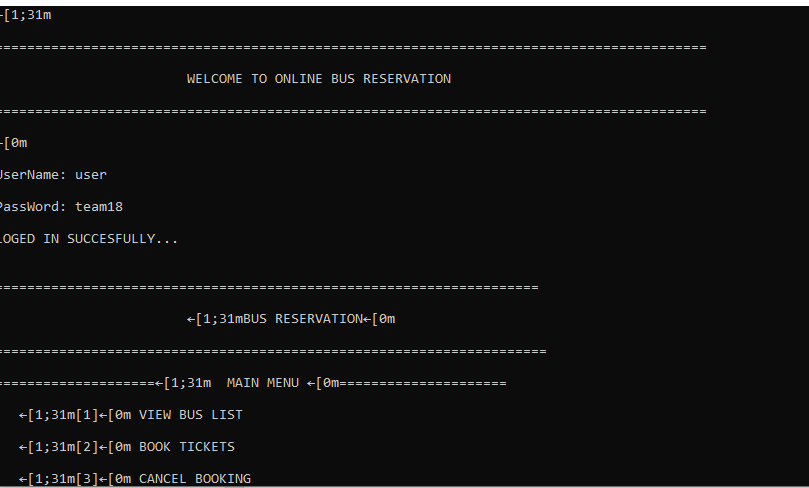
printf("\n\n");

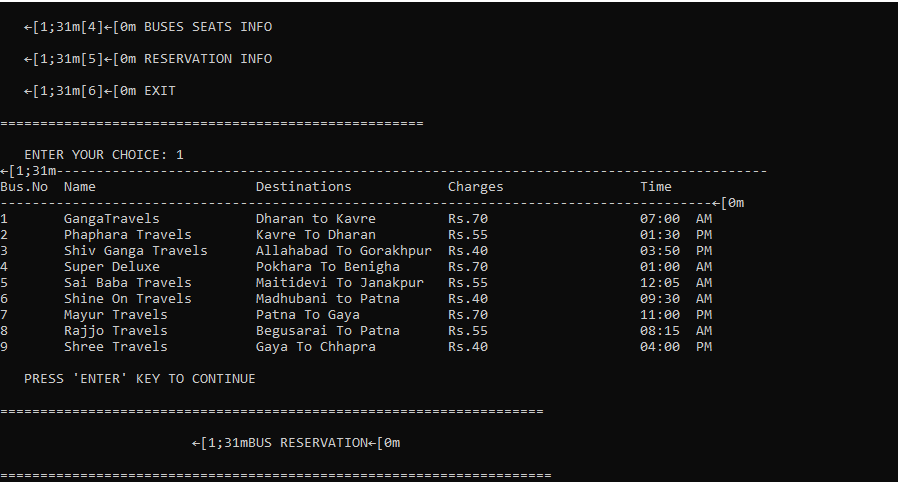
getch();

return 0;

}

}





Calendar:

#include <stdio.h>

int dayOfWeek(int d, int m, int y) {

static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};

y -= m < 3;

return (y + y/4 - y/100 + y/400 + t[m-1] + d) % 7;

}

void printCalendar(int month, int year) {

int daysInMonth, i, currentDay;

int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))

days[1] = 29;

printf(" \*\*\*\* Calendar - %d/%d \*\*\*\*\n", month, year);

printf(" Sun Mon Tue Wed Thu Fri Sat\n");

currentDay = dayOfWeek(1, month, year);

for (i = 0; i < currentDay; i++)

printf(" ");

for (i = 1; i <= days[month-1]; i++) {

printf("%5d", i);

if (++currentDay > 6) {

currentDay = 0;

printf("\n");

}

}

if (currentDay != 0)

printf("\n");

}

int main() {

int month, year;

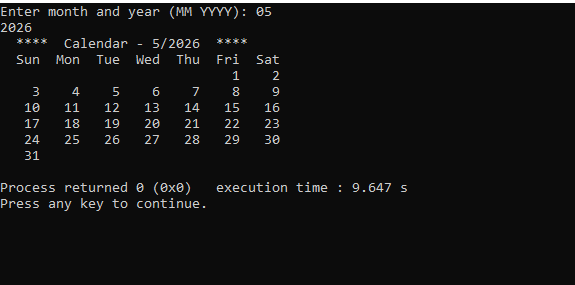
printf("Enter month and year (MM YYYY): ");

scanf("%d %d", &month, &year);

printCalendar(month, year);

return 0;

}



Conversions:

#include<stdio.h>

#include<string.h>

#include<math.h>

void start();

void convert();

void binary\_Values();

int bin\_to\_dec();

void bin\_to\_oct();

void bin\_to\_hexa();

void decimal\_values();

void dec\_to\_binary();

void dec\_to\_oct();

void dec\_to\_hexa();

void octal\_values();

void oct\_bin();

void hexa\_values();

void hexa\_bin();

void wish();

void wish(){

wis:

printf("----------------------------------\n");

printf("Do you wish to Continue?\n");

printf("Press Y to continue or press N to cancel.\n");

printf("----------------------------------\n");

char c;

scanf("%c", &c);

if (c == 'Y' || c == 'y')

{

system("cls");

start();

}

else if(c=='N' || c == 'n') {

printf("Exiting...\n");

return ;

}

else{

printf("Please enter a valid Keyword.\n");

goto wis;

}

}

void start(){

start:

printf("press 1 for Binary Conversions\n");

printf("Press 2 for Decimal Conversions\n");

printf("Press 3 for Octal Conversions\n");

printf("Press 4 for HexDecimal Conversions\n");

printf("Press 5 for Exit Conversions\n");

printf("----------------------------------\n");

int option;

scanf("%d", &option);

if(option ==5){

printf("Exiting.....!\n");

return ;

}

else if(option>4){

printf("Choose the correct option\n");

printf("----------------------------------\n");

goto start;

}

else{

convert(option);

}

}

void convert(int a){

switch (a)

{

case 1:

binary\_Values();

wish();

break;

case 2:

decimal\_values();

wish();

break;

case 3:

octal\_values();

wish();

break;

case 4:

hexa\_values();

wish();

break;

default:

break;

}

}

void binary\_Values(){

bin:

printf("Please Enter the Binary Value\n");

char a[20];

scanf("%s",a);

for(int i=0;i<20;i++)

{

if(!a[i]==0 && !a[i]==1){

printf("Enter the Binary Values only\n");

goto bin;

}

}

printf("Decimal value of %s is : %d\n",a,bin\_to\_dec(a));

bin\_to\_oct(a);

bin\_to\_hexa(a);

}

int bin\_to\_dec(char a[]){

int dec=0;

int c=0;

for(int i=strlen(a)-1;i>=0;i--){

dec=dec+(a[i]-'0')\*(int)pow(2,c);

c++;

}

return dec;

}

void bin\_to\_oct(char a[]){

int dec=bin\_to\_dec(a);

char a1[20];

sprintf(a1,"%o",dec);

printf("Octal value of %s is : %s\n",a,a1);

}

void bin\_to\_hexa(char a[]){

int dec=bin\_to\_dec(a);

char a1[20];

sprintf(a1,"%x",dec);

printf("Hexa value of %s is: %s\n",a,a1);

}

void decimal\_values(){

printf("Enter a decimal value:\n");

int i;

scanf("%d",&i);

if(!i>0){

printf("Enter a decimal value\n");

}

dec\_to\_binary(i);

dec\_to\_oct(i);

dec\_to\_hexa(i);

}

void dec\_to\_binary(int a){

int c=0;

int bin[20];

while(a>0){

bin[c++]=a%2;

a=a/2;

}

printf("Binary value of %d is :",a);

for(int i=c-1;i>=0;i--){

printf("%d",bin[i]);

}

printf("\n");

}

void dec\_to\_oct(int a){

char dec[20];

sprintf(dec, "%o",a);

printf("Octal value of %d is :%s\n",a,dec);

}

void dec\_to\_hexa(int a){

char hex[20];

sprintf(hex, "%x",a);

printf("Hexadecimal value of %d is :%s\n",a,hex);

}

void octal\_values(){

printf("Enter your Octal Number:\n");

char i[20];

scanf("%s",i);

oct\_bin(i);

}

void oct\_bin(char a[]){

int dec=0;

int c=0;

for(int i=strlen(a)-1;i>=0;i--){

dec=dec+(a[i]-'0')\*(int)pow(8,c);

c++;

}

printf("Decimal Value of %s is :%d\n",a,dec);

dec\_to\_binary(dec);

dec\_to\_hexa(dec);

}

void hexa\_values(){

hexa:

printf("Enter the hexadecimal Values\n");

char c[20];

scanf("%s",c);

for(int i=0;i<strlen(c);i++){

if((c[i]>='0' && c[i]<='9') || (c[i]>='a' && c[i]<='f')){

}

else{

printf("Invalid hexadecimal value\n");

// system("cls");

goto hexa;

}

}

hexa\_bin(c);

}

void hexa\_bin(char a[]){

int dec=0;

int c=0;

for(int i=strlen(a)-1;i>=0;i--){

if((a[i]>='0' && a[i]<='9')){

dec=dec+(a[i]-'0')\*(int)pow(16,c);

}

else{

dec=dec+(a[i]-'a'+10)\*(int)pow(16,c);

}

c++;

}

printf("Decimal Value of %s is: %d\n",a,dec);

dec\_to\_binary(dec);

dec\_to\_oct(dec);

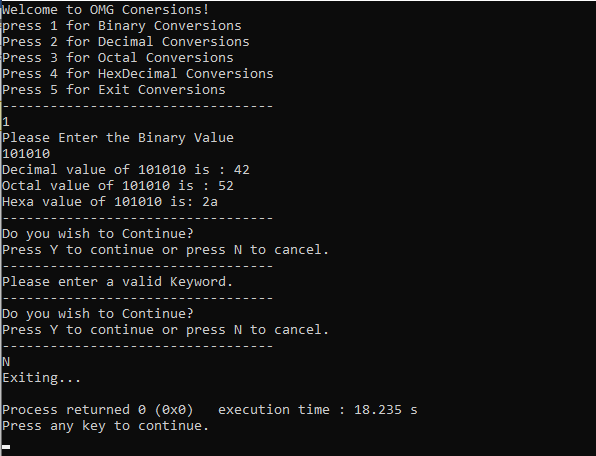
}

int main(){

printf("Welcome to OMG Conersions!\n");

start();

}



Double Linked List:

#include <stdio.h>

#include <stdlib.h>

// Define the structure of a doubly linked list node

typedef struct Node {

int data;

struct Node\* next;

struct Node\* prev;

} Node;

// Function to create a new node

Node\* createNode(int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

if (!newNode) {

printf("Memory error\n");

return NULL;

}

newNode->data = data;

newNode->next = NULL;

newNode->prev = NULL;

return newNode;

}

// Function to append a node to the end of the list

void appendNode(Node\*\* head, int data) {

Node\* newNode = createNode(data);

if (!newNode) return;

if (!\*head) {

\*head = newNode;

return;

}

Node\* temp = \*head;

while (temp->next) {

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

}

// Function to insert a node at the beginning of the list

void insertAtBeginning(Node\*\* head, int data) {

Node\* newNode = createNode(data);

if (!newNode) return;

if (!\*head) {

\*head = newNode;

return;

}

newNode->next = \*head;

(\*head)->prev = newNode;

\*head = newNode;

}

// Function to delete a node from the list

void deleteNode(Node\*\* head, int key) {

if (!\*head) {

printf("List is empty\n");

return;

}

Node\* temp = \*head;

while (temp && temp->data != key) {

temp = temp->next;

}

if (!temp) {

printf("Node with data %d not found\n", key);

return;

}

if (temp->prev) {

temp->prev->next = temp->next;

} else {

\*head = temp->next;

}

if (temp->next) {

temp->next->prev = temp->prev;

}

free(temp);

}

// Function to display the list in forward direction

void displayListForward(Node\* head) {

if (!head) {

printf("List is empty\n");

return;

}

Node\* temp = head;

printf("Forward: ");

while (temp) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

// Function to display the list in reverse direction

void displayListReverse(Node\* head) {

if (!head) {

printf("List is empty\n");

return;

}

Node\* temp = head;

while (temp->next) {

temp = temp->next;

}

printf("Reverse: ");

while (temp) {

printf("%d ", temp->data);

temp = temp->prev;

}

printf("\n");

}

// Function to free all nodes in the list

void freeList(Node\*\* head) {

Node\* temp = \*head;

Node\* next = NULL;

while (temp) {

next = temp->next;

free(temp);

temp = next;

}

\*head = NULL;

}

// Function to display the menu

void displayMenu() {

printf("Menu:\n");

printf("1. Append node\n");

printf("2. Insert node at beginning\n");

printf("3. Delete node\n");

printf("4. Display list forward\n");

printf("5. Display list reverse\n");

printf("6. Exit\n");

}

int main() {

Node\* head = NULL;

int choice, data;

while (1) {

displayMenu();

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data to append: ");

scanf("%d", &data);

appendNode(&head, data);

break;

case 2:

printf("Enter data to insert at beginning: ");

scanf("%d", &data);

insertAtBeginning(&head, data);

break;

case 3:

printf("Enter data to delete: ");

scanf("%d", &data);

deleteNode(&head, data);

break;

case 4:

displayListForward(head);

break;

case 5:

displayListReverse(head);

break;

case 6:

freeList(&head);

printf("Exiting program.\n");

return 0;

default:

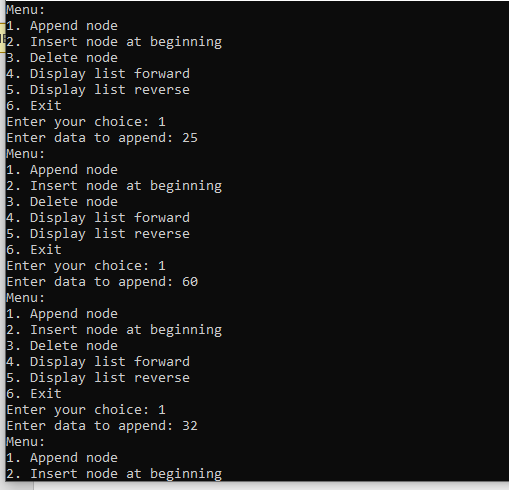
printf("Invalid choice. Please try again.\n");

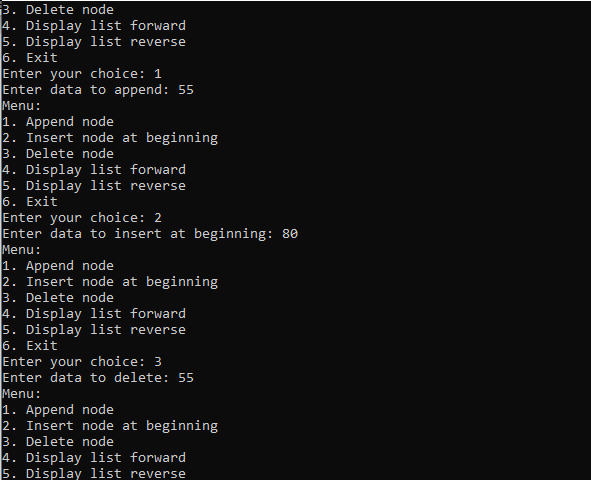
}

}

return 0;

}





## 

Employee Details:

#include <stdio.h>

#include <string.h>

struct emp

{

char name[30];

int age;

char branch[20];

char designation[20];

};

void main()

{

struct emp p1;

printf("Enter Employee name: \n");

scanf("%s", p1.name);

strcpy (p1.name, p1.name);

printf("Enter Employee Age: \n");

scanf("%d", &p1.age);

printf("Enter Employee Branch:\n");

scanf("%s", p1.branch);

printf("Enter Employee Designation:\n");

scanf("%s", p1.designation);

FILE \*fp=fopen("D:\\Sumayya\\Demo.txt", "a");

if(fp==NULL){

printf("Error opening File\n");

return ;

}

fprintf(fp,"Employee Name:%s\n Employee Age: %d \n Employee Branch:%s\n Employee Designation :%s\n ",p1.name,p1.age,p1.branch,p1.designation);

printf("Data Uploaded Successfully\n");

printf("Entered Details\n");

printf("Employee Name:%s\n",p1.name);

printf("Employee Branch:%s\n",p1.branch);

printf("Employee Designation:%s\n",p1.designation);

printf("Employee Age %d",p1.age);

FILE \*ptr;

ptr=fopen("D:\\Sumayya\\Demo.txt", "r");

if(ptr==NULL){

printf("Failed to open File\n");

return;

}

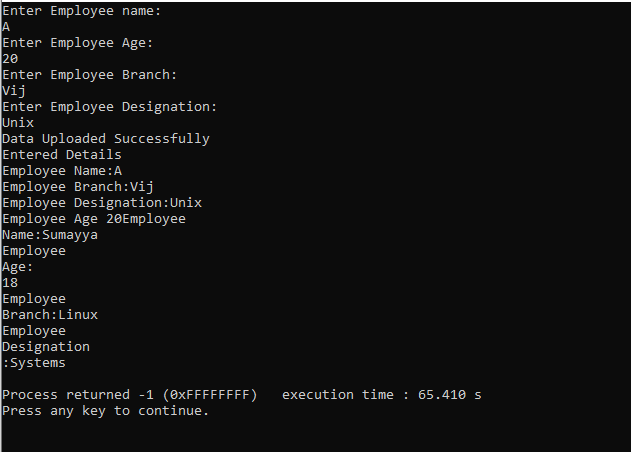
char a[100];

while(fscanf(ptr,"%s",a)!=EOF){

printf("%s\n",a);

}

}



Employee Record:

#include <stdio.h>

#include <string.h>

struct emp {

char name[30];

char department[30];

int age;

} p1;

int main() {

strcpy(p1.name, "Sumayya Farheen");

strcpy(p1.department, "Engineering");

p1.age = 30;

FILE \*filePtr;

filePtr = fopen("D:\\Sumayya\\employee\_record.txt", "w");

if (filePtr == NULL) {

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

return 1;

}

fprintf(filePtr, "Employee Record\n");

fprintf(filePtr, "Name: %s\n", p1.name);

fprintf(filePtr, "Department: %s\n", p1.department);

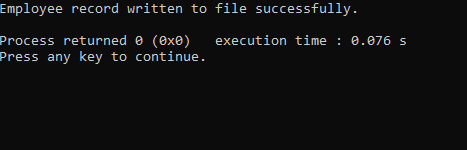
fprintf(filePtr, "Age: %d\n", p1.age);

fclose(filePtr);

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

return 0;

}



Employee Management System:

#include <stdio.h>

#include <string.h>

void create();

void view();

void edit();

void delete();

void choice();

void choice(){

int choic;

printf("\nWELCOME TO OMG COMPANY\n");

start:

printf("--------------------------------\n");

printf("Press 1 to Create a new Data\n");

printf("Press 2 to View the Data\n");

printf("Press 3 to edit the Data\n");

printf("Press 4 to delete the Data\n");

printf("Press 5 to exit the Program...!\n");

printf("--------------------------------\n");

int choice;

scanf("%d",&choice);

if(choice==5){

printf("Exiting the Program...\n");

return ;

}

switch (choice)

{

case 1:

create();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

case 2:

view();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

case 3:

edit();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

case 4:

delete();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

default:

break;

}

}

struct emp

{

int id;

char name[30];

int age;

char branch[20];

char designation[20];

}emp;

void create(){

struct emp p1;

FILE \*fp=fopen("D:\\Assignments\\Demo1.txt", "a");

if(fp==NULL){

printf("Error opening File");

return ;

}

printf("\nEnter Employee Id:");

scanf("%d",&p1.id);

// fflush(stdin);

printf("\nEnter Employee name: ");

scanf("%s", p1.name);

strcpy (p1.name, p1.name);

printf("\nEnter Employee Age: ");

scanf("%d", &p1.age);

printf("\nEnter Employee Branch:");

scanf("%s", p1.branch);

printf("\nEnter Employee Designation:");

scanf("%s", p1.designation);

//fprintf(fp,"Employee Id:%d\n Employee Name:%s\n Employee Age: %d \n Employee Branch:%s\n Employee Designation :%s\n\n ",p1.id,p1.name,p1.age,p1.branch,p1.designation);

fwrite(&p1,sizeof(emp),1,fp);

printf("Data Uploaded Successfully\n");

fclose(fp);

}

void view(){

struct emp p1;

FILE \*ptr;

ptr=fopen("D:\\Assignments\\Demo1.txt", "r");

if(ptr==NULL){

printf("Failed to open File\n");

return;

}

while(fread(&p1,sizeof(emp),1,ptr)){

printf("%d %s %d %s %s\n",p1.id,p1.name,p1.age,p1.branch,p1.designation);

}

fclose(ptr);

}

void edit(){

struct emp p1;

FILE \*ptr, \*ptr1;

int c = 0;

int a;

ptr = fopen("D:\\Assignments\\Demo1.txt", "r");

if (ptr == NULL) {

printf("Error opening file for reading\n");

return ;

}

ptr1 = fopen("D:\\Assignments\\temp.txt", "w");

if (ptr1 == NULL) {

printf("Error opening file for writing\n");

fclose(ptr);

return ;

}

printf("Enter the Employee Id to Modify the Data:\n");

scanf("%d", &a);

while (fread(&p1, sizeof(struct emp), 1, ptr)) {

if (a == p1.id) {

c = 1;

printf("Enter New Employee name: \n");

scanf("%s", p1.name);

printf("Enter New Employee Age: \n");

scanf("%d", &p1.age);

printf("Enter New Employee Branch:\n");

scanf("%s", p1.branch);

printf("Enter New Employee Designation:\n");

scanf("%s", p1.designation);

}

fwrite(&p1, sizeof(struct emp), 1, ptr1);

}

fclose(ptr);

fclose(ptr1);

if (c == 0) {

printf("Employee Not found\n");

} else {

ptr1 = fopen("D:\\Assignments\\temp.txt", "r");

ptr = fopen("D:\\Assignments\\Demo1.txt", "w");

if (ptr == NULL) {

printf("Error opening file for writing\n");

fclose(ptr1);

return ;

}

if (ptr1 == NULL) {

printf("Error opening file for reading\n");

fclose(ptr);

return ;

}

while (fread(&p1, sizeof(struct emp), 1, ptr1)) {

fwrite(&p1, sizeof(struct emp), 1, ptr);

}

fclose(ptr);

fclose(ptr1);

remove("D:\\Assignments\\temp.txt");

}

}

void delete(){

FILE \*ptr, \*ptr1;

struct emp p1;

ptr = fopen("D:\\Assignments\\Demo.txt", "r");

ptr1 = fopen("D:\\Assignments\\temp.txt", "w");

if(ptr == NULL){

printf("Error opening files for deleting\n");

fclose(ptr);

return;

}

if(ptr1==NULL){

printf("Error opening files for deleting\n");

fclose(ptr1);

return ;

}

printf("Enter the employee id to delete\n");

int a;

scanf("%d", &a);

int c = 0;

while (fread(&p1, sizeof(struct emp), 1, ptr)){

printf("%d",p1.id);

if(a == p1.id){

c = 1;

} else {

fwrite(&p1, sizeof(struct emp), 1, ptr1);

}

}

fclose(ptr1);

fclose(ptr);

if(c == 1){

ptr = fopen("D:\\Assignments\\temp.txt", "r");

ptr1 = fopen("D:\\Assignments\\Demo1.txt", "w");

if(ptr == NULL || ptr1 == NULL){

printf("Failed to open the file\n");

if(ptr) fclose(ptr);

if(ptr1) fclose(ptr1);

return;

}

while(fread(&p1, sizeof(struct emp), 1, ptr)){

fwrite(&p1, sizeof(struct emp), 1, ptr1);

}

fclose(ptr1);

fclose(ptr);

remove("D:\\Assignments\\temp.txt");

printf("Deleted the employee successfully\n");

} else {

printf("Record not found\n");

}

}

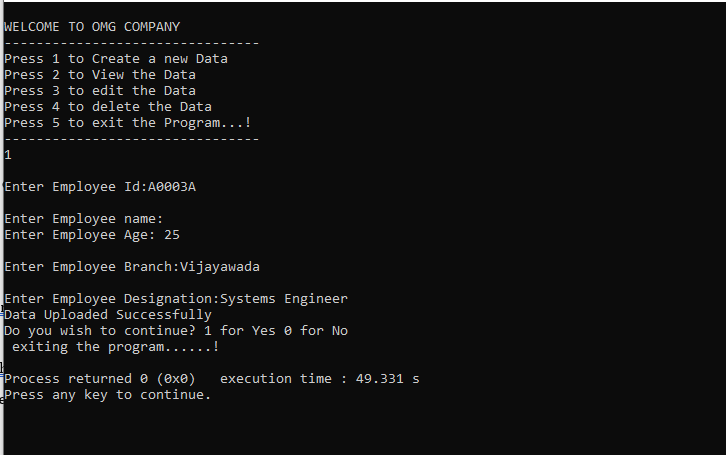
int main()

{

choice();

return 0;

}



File Handling:

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

void main ()

{

int n;

FILE \*fptr;

fptr = fopen ("D:\\Sumayya\\Hello.txt", "a");

if (fptr == NULL)

{

printf ("Error!!!!!");

exit(0);

}

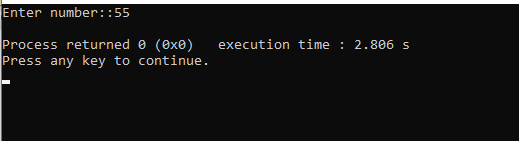
printf ("Enter number::");

scanf ("%d", &n);

fprintf (fptr, "%d", n);

fclose (fptr);

}



File Operations:

#include <stdio.h> ///for input output functions like printf, scanf

#include <stdlib.h>

#include <conio.h>

#include <windows.h> ///for windows related functions (not important)

#include <string.h> ///string operations

/\*\* List of Global Variable \*/

COORD coord = {0,0}; /// top-left corner of window

/\*\*

function : gotoxy

@param input: x and y coordinates

@param output: moves the cursor in specified position of console

\*/

void gotoxy(int x,int y)

{

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

/\*\* Main function started \*/

int main()

{

FILE \*fp, \*ft; /// file pointers

char another, choice;

/\*\* structure that represent a employee \*/

struct emp

{

char name[40]; ///name of employee

int age; /// age of employee

float bs; /// basic salary of employee

};

struct emp e; /// structure variable creation

char empname[40]; /// string to store name of the employee

long int recsize; /// size of each record of employee

/\*\* open the file in binary read and write mode

\* if the file EMP.DAT already exists then it open that file in read write mode

\* if the file doesn't exit it simply create a new copy

\*/

fp = fopen("EMP.DAT","rb+");

if(fp == NULL)

{

fp = fopen("EMP.DAT","wb+");

if(fp == NULL)

{

printf("Connot open file");

exit(1);

}

}

/// sizeo of each record i.e. size of structure variable e

recsize = sizeof(e);

/// infinite loop continues untile the break statement encounter

while(1)

{

system("cls"); ///clear the console window

gotoxy(30,10); /// move the cursor to postion 30, 10 from top-left corner

printf("1. Add Record"); /// option for add record

gotoxy(30,12);

printf("2. List Records"); /// option for showing existing record

gotoxy(30,14);

printf("3. Modify Records"); /// option for editing record

gotoxy(30,16);

printf("4. Delete Records"); /// option for deleting record

gotoxy(30,18);

printf("5. Exit"); /// exit from the program

gotoxy(30,20);

printf("Your Choice: "); /// enter the choice 1, 2, 3, 4, 5

fflush(stdin); /// flush the input buffer

choice = getche(); /// get the input from keyboard

switch(choice)

{

case '1': /// if user press 1

system("cls");

fseek(fp,0,SEEK\_END); /// search the file and move cursor to end of the file

/// here 0 indicates moving 0 distance from the end of the file

another = 'y';

while(another == 'y') /// if user want to add another record

{

printf("\nEnter name: ");

scanf("%s",e.name);

printf("\nEnter age: ");

scanf("%d", &e.age);

printf("\nEnter basic salary: ");

scanf("%f", &e.bs);

fwrite(&e,recsize,1,fp); /// write the record in the file

printf("\nAdd another record(y/n) ");

fflush(stdin);

another = getche();

}

break;

case '2':

system("cls");

rewind(fp); ///this moves file cursor to start of the file

while(fread(&e,recsize,1,fp)==1) /// read the file and fetch the record one record per fetch

{

printf("\n%s %d %.2f",e.name,e.age,e.bs); /// print the name, age and basic salary

}

getch();

break;

case '3': /// if user press 3 then do editing existing record

system("cls");

another = 'y';

while(another == 'y')

{

printf("Enter the employee name to modify: ");

scanf("%s", empname);

rewind(fp);

while(fread(&e,recsize,1,fp)==1) /// fetch all record from file

{

if(strcmp(e.name,empname) == 0) ///if entered name matches with that in file

{

printf("\nEnter new name,age and bs: ");

scanf("%s%d%f",e.name,&e.age,&e.bs);

fseek(fp,-recsize,SEEK\_CUR); /// move the cursor 1 step back from current position

fwrite(&e,recsize,1,fp); /// override the record

break;

}

}

printf("\nModify another record(y/n)");

fflush(stdin);

another = getche();

}

break;

case '4':

system("cls");

another = 'y';

while(another == 'y')

{

printf("\nEnter name of employee to delete: ");

scanf("%s",empname);

ft = fopen("Temp.dat","wb"); /// create a intermediate file for temporary storage

rewind(fp); /// move record to starting of file

while(fread(&e,recsize,1,fp) == 1) /// read all records from file

{

if(strcmp(e.name,empname) != 0) /// if the entered record match

{

fwrite(&e,recsize,1,ft); /// move all records except the one that is to be deleted to temp file

}

}

fclose(fp);

fclose(ft);

remove("EMP.DAT"); /// remove the orginal file

rename("Temp.dat","EMP.DAT"); /// rename the temp file to original file name

fp = fopen("EMP.DAT", "rb+");

printf("Delete another record(y/n)");

fflush(stdin);

another = getche();

}

break;

case '5':

fclose(fp); /// close the file

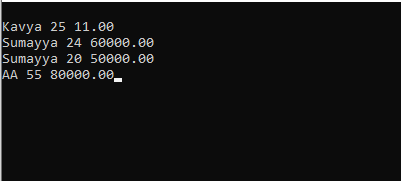
exit(0); /// exit from the program

}

}

return 0;

}



Linked List:

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

struct node {

char name[20];

int age;

struct node \*next;

};

void printLinkedlist(struct node \*p) {

while (p != NULL) {

printf("\n%s",p->name);

printf("\n%d ",p->age);

p = p->next;

}

}

int main() {

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

strcpy(one->name,"Sumayya");

one->age = 1;

strcpy(two->name,"Hanee");

two->age = 2;

strcpy(three->name,"Mahamuda");

three->age = 3;

one->next = two;

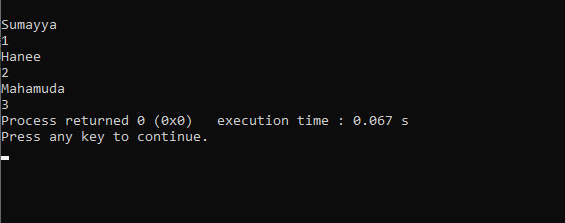
two->next = three;

three->next = NULL;

head = one;

printLinkedlist(head);

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Node {

char name[20];

int age;

struct Node \*next;

};

struct Node \*head;

void insert(char name[], int age) {

struct Node \*h = head;

struct Node \*nn = (struct Node\*)malloc(sizeof(struct Node));

nn->age = age;

strcpy(nn->name, name);

nn->next = NULL;

if (h == NULL) {

head = nn;

return;

}

while (h->next != NULL) {

h = h->next;

}

h->next = nn;

}

void display() {

printf("Checking..\n");

struct Node\* h = head;

while (h != NULL) {

printf("\nName: %s and Age: %d\n", h->name, h->age);

h = h->next;

}

}

int main() {

printf("Enter the number of nodes you want to create:\n");

int n;

scanf("%d", &n);

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

int age;

char name[20];

printf("Enter age for node %d\n: ", i);

scanf("%d", &age);

printf("Enter name for node %d\n: ", i);

scanf("%s", name);

insert(name, age);

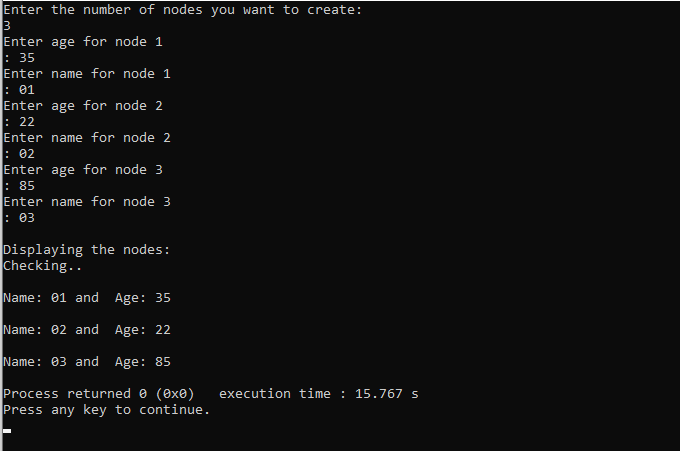
}

printf("\nDisplaying the nodes:\n");

display();

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

struct node{

int data;

struct node \*link;

};

void insert(struct node \*\*head, int data)

{

struct node \*newnode = (struct node \*) malloc (sizeof (struct node));

newnode->data = data;

newnode->link = \*head;

\*head = newnode ;

}

void display (struct node \*Node)

{

while (Node != NULL)

{

printf ("%d\t", Node->data);

Node = Node->link;

}

printf("\n");

}

main()

{

struct node \*head = NULL;

struct node \*node2 = NULL;

struct node \*node3 = NULL;

head = (struct node \*) malloc (sizeof (struct node));

node2 = (struct node \*) malloc (sizeof (struct node));

node3 = (struct node \*) malloc (sizeof (struct node));

head->data = 9;

head->link = node2;

node2->data = 10;

node2->link = node3;

node3->data = 11;

node3->link = NULL;

printf("Elements are:: \n");

display (head);

insert(&head, 12);

printf ("After inserting ::\n");

display (head);

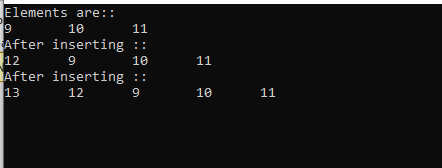
insert(&head, 13);

printf ("After inserting ::\n");

display (head);

getch();

}



Stack:

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int count = 0;

struct stack {

int items[MAX];

int top;

};

typedef struct stack st;

void createEmptyStack(st \*s) {

s->top = -1;

}

int isfull(st \*s) {

if (s->top == MAX - 1)

return 1;

else

return 0;

}

int isempty(st \*s) {

if (s->top == -1)

return 1;

else

return 0;

}

void push(st \*s, int newitem) {

if (isfull(s)) {

printf("STACK FULL");

} else {

s->top++;

s->items[s->top] = newitem;

}

count++;

}

void pop(st \*s) {

if (isempty(s)) {

printf("\n STACK EMPTY \n");

} else {

printf("Item popped= %d", s->items[s->top]);

s->top--;

}

count--;

printf("\n");

}

void printStack(st \*s) {

printf("Stack: ");

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

printf("%d ", s->items[i]);

}

printf("\n");

}

int main() {

int ch;

st \*s = (st \*)malloc(sizeof(st));

createEmptyStack(s);

push(s, 1);

push(s, 2);

push(s, 3);

push(s, 4);

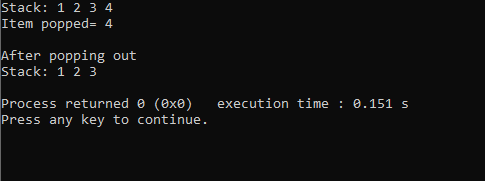
printStack(s);

pop(s);

printf("\nAfter popping out\n");

printStack(s);

}



String:

#include <stdio.h>

#include <conio.h>

main()

{

char name[30];

printf("\nEnter Your name ::");

//scanf ("%s", name);

fgets(name, sizeof(name), stdin);

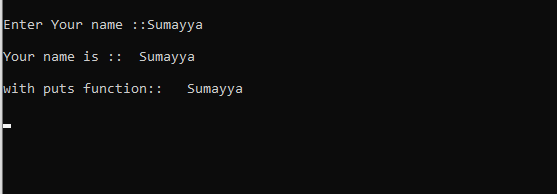
printf("\nYour name is :: %s", name);

printf("\nwith puts function:: ");

puts(name);

getch();

}



Structures:

#include <stdio.h>

struct person

{

int age;

float weight;

};

int main(){

struct person \*personPtr, person1; personPtr = &person1; printf("Enter age: ");

scanf("%d", &personPtr->age);

printf("Enter weight: ");

scanf("%f", &personPtr->weight);

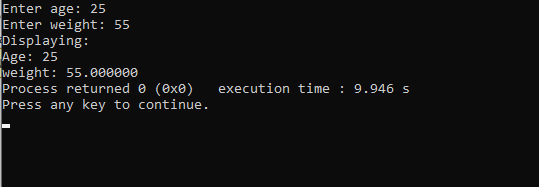
printf("Displaying:\n");

printf("Age: %d\n", personPtr->age);

printf("weight: %f", personPtr->weight);

return 0;

}



#include <stdio.h>

#include <stdlib.h>

struct person {

int age;

float weight;

char name[30];

};

int main()

{

struct person \*ptr;

int i, n;

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

scanf("%d", &n);

// allocating memory for n numbers of struct person

ptr = (struct person\*) malloc(n \* sizeof(struct person));

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

{

printf("Enter first name and age respectively: ");

// To access members of 1st struct person,

// ptr->name and ptr->age is used

// To access members of 2nd struct person,

// (ptr+1)->name and (ptr+1)->age is used

scanf("%s %d", (ptr+i)->name, &(ptr+i)->age);

}

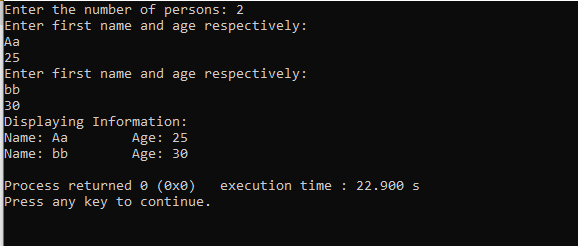
printf("Displaying Information:\n");

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

printf("Name: %s\tAge: %d\n", (ptr+i)->name, (ptr+i)->age);

return 0;

}



#include <stdio.h>

struct student {

char name[50];

int age;

};

// function prototype

void display(struct student s);

int main() {

struct student s1;

printf("Enter name: ");

// read string input from the user until \n is entered

// \n is discarded

scanf("%[^\n]%\*c", s1.name);

printf("Enter age: ");

scanf("%d", &s1.age);

display(s1); // passing struct as an argument

return 0;

}

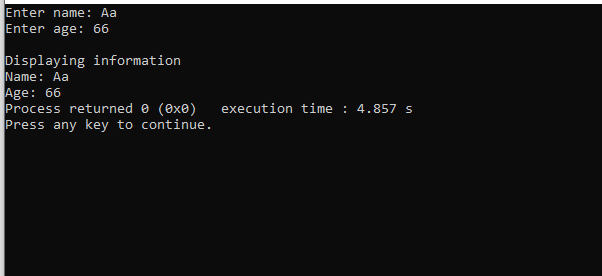
void display(struct student s) {

printf("\nDisplaying information\n");

printf("Name: %s", s.name);

printf("\nAge: %d", s.age);

}



// AVL tree implementation in C

#include <stdio.h>

#include <stdlib.h>

// Create Node

struct Node {

int key;

struct Node \*left;

struct Node \*right;

int height;

};

int max(int a, int b);

// Calculate height

int height(struct Node \*N) {

if (N == NULL)

return 0;

return N->height;

}

int max(int a, int b) {

return (a > b) ? a : b;

}

// Create a node

struct Node \*newNode(int key) {

struct Node \*node = (struct Node \*)

malloc(sizeof(struct Node));

node->key = key;

node->left = NULL;

node->right = NULL;

node->height = 1;

return (node);

}

// Right rotate

struct Node \*rightRotate(struct Node \*y) {

struct Node \*x = y->left;

struct Node \*T2 = x->right;

x->right = y;

y->left = T2;

y->height = max(height(y->left), height(y->right)) + 1;

x->height = max(height(x->left), height(x->right)) + 1;

return x;

}

// Left rotate

struct Node \*leftRotate(struct Node \*x) {

struct Node \*y = x->right;

struct Node \*T2 = y->left;

y->left = x;

x->right = T2;

x->height = max(height(x->left), height(x->right)) + 1;

y->height = max(height(y->left), height(y->right)) + 1;

return y;

}

// Get the balance factor

int getBalance(struct Node \*N) {

if (N == NULL)

return 0;

return height(N->left) - height(N->right);

}

// Insert node

struct Node \*insertNode(struct Node \*node, int key) {

// Find the correct position to insertNode the node and insertNode it

if (node == NULL)

return (newNode(key));

if (key < node->key)

node->left = insertNode(node->left, key);

else if (key > node->key)

node->right = insertNode(node->right, key);

else

return node;

// Update the balance factor of each node and

// Balance the tree

node->height = 1 + max(height(node->left),

height(node->right));

int balance = getBalance(node);

if (balance > 1 && key < node->left->key)

return rightRotate(node);

if (balance < -1 && key > node->right->key)

return leftRotate(node);

if (balance > 1 && key > node->left->key) {

node->left = leftRotate(node->left);

return rightRotate(node);

}

if (balance < -1 && key < node->right->key) {

node->right = rightRotate(node->right);

return leftRotate(node);

}

return node;

}

struct Node \*minValueNode(struct Node \*node) {

struct Node \*current = node;

while (current->left != NULL)

current = current->left;

return current;

}

// Delete a nodes

struct Node \*deleteNode(struct Node \*root, int key) {

// Find the node and delete it

if (root == NULL)

return root;

if (key < root->key)

root->left = deleteNode(root->left, key);

else if (key > root->key)

root->right = deleteNode(root->right, key);

else {

if ((root->left == NULL) || (root->right == NULL)) {

struct Node \*temp = root->left ? root->left : root->right;

if (temp == NULL) {

temp = root;

root = NULL;

} else

\*root = \*temp;

free(temp);

} else {

struct Node \*temp = minValueNode(root->right);

root->key = temp->key;

root->right = deleteNode(root->right, temp->key);

}

}

if (root == NULL)

return root;

// Update the balance factor of each node and

// balance the tree

root->height = 1 + max(height(root->left),

height(root->right));

int balance = getBalance(root);

if (balance > 1 && getBalance(root->left) >= 0)

return rightRotate(root);

if (balance > 1 && getBalance(root->left) < 0) {

root->left = leftRotate(root->left);

return rightRotate(root);

}

if (balance < -1 && getBalance(root->right) <= 0)

return leftRotate(root);

if (balance < -1 && getBalance(root->right) > 0) {

root->right = rightRotate(root->right);

return leftRotate(root);

}

return root;

}

// Print the tree

void printPreOrder(struct Node \*root) {

if (root != NULL) {

printf("%d ", root->key);

printPreOrder(root->left);

printPreOrder(root->right);

}

}

int main() {

struct Node \*root = NULL;

root = insertNode(root, 2);

root = insertNode(root, 1);

root = insertNode(root, 7);

root = insertNode(root, 4);

root = insertNode(root, 5);

root = insertNode(root, 3);

root = insertNode(root, 8);

printPreOrder(root);

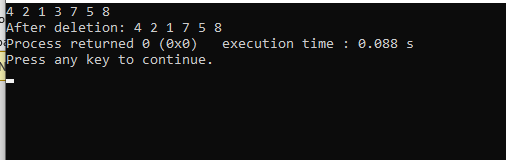
root = deleteNode(root, 3);

printf("\nAfter deletion: ");

printPreOrder(root);

return 0;

}



Prime:

#include<stdio.h>

int main(){

int a;

scanf("%d",&a);

prime(a);

}

void prime(int a){

if(a<=1){

printf("%d is not prime",a);

return ;

}

int c=0;

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

if(a%i==0){

c++;

}

}

if(c==2){

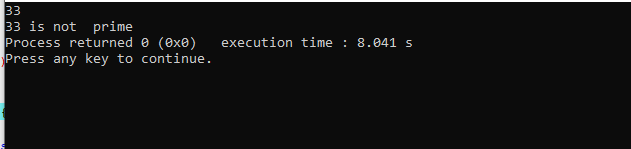
printf("%d is prime",a);

}

else

printf("%d is not prime",a);

}



struct size:

#include <stdio.h>

struct name {

int member1;

int member2;

};

int main()

{

struct name \*ptr, Harry;

printf("Size of struct name: %zu bytes\n", sizeof(struct name));

printf("Size of member1: %zu bytes\n", sizeof(Harry.member1));

printf("Size of member2: %zu bytes\n", sizeof(Harry.member2));

ptr=&Harry;

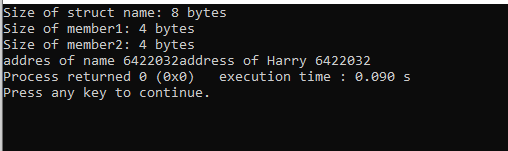
printf("addres of name %zu",ptr);

ptr=&Harry.member1;

printf("address of Harry %zu",ptr);

return 0;

}



Union:

#include <stdio.h>union Job {

float salary;

int workerNo;

} j;

int main() {

j.salary = 12.3;

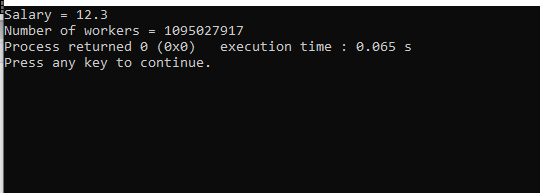
// when j.workerNo is assigned a value,// j.salary will no longer hold 12.3 j.workerNo = 100;

printf("Salary = %.1f\n", j.salary);

printf("Number of workers = %d", j.workerNo);

return 0;

}



Add:

#include <stdio.h>

int sum(int n);

int main() {

int number, result;

printf("Enter a positive integer: ");

scanf("%d", &number);

result = sum(number);

printf("sum = %d", result);

return 0;

}

int sum(int n) {

if (n != 0)

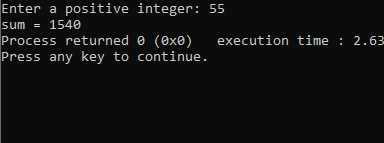
// sum() function calls itself

return n + sum(n-1);

else

return n;

}



Day:10

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

typedef struct Treenode {

int data;

struct Treenode \*left, \*right;

} Treenode;

typedef struct {

Treenode \*root;

} Tree;

Treenode\* newTreenode(int data) {

Treenode\* node = (Treenode\*)malloc(sizeof(Treenode));

node->data = data;

node->left = node->right = NULL;

return node;

}

int height(Treenode \*root) {

if (root == NULL)

return 0;

int left\_height = height(root->left);

int right\_height = height(root->right);

return (left\_height > right\_height ? left\_height : right\_height) + 1;

}

int getcol(int h) {

if (h == 1)

return 1;

return getcol(h - 1) + getcol(h - 1) + 1;

}

void printTree(int \*\*M, Treenode \*root, int col, int row, int height) {

if (root == NULL)

return;

M[row][col] = root->data;

printTree(M, root->left, col - pow(2, height - 2), row + 1, height - 1);

printTree(M, root->right, col + pow(2, height - 2), row + 1, height - 1);

}

void TreePrinter(Tree tree) {

int h = height(tree.root);

int col = getcol(h);

int \*\*M = (int \*\*)malloc(h \* sizeof(int \*));

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

M[i] = (int \*)malloc(col \* sizeof(int));

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

M[i][j] = 0;

}

}

printTree(M, tree.root, col / 2, 0, h);

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

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

if (M[i][j] == 0)

printf(" ");

else

printf("%d ", M[i][j]);

}

printf("\n");

}

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

free(M[i]);

}

free(M);

}

Treenode\* insertLevelOrder(int arr[], Treenode\* root, int i, int n) {

if (i < n) {

Treenode \*temp = newTreenode(arr[i]);

root = temp;

root->left = insertLevelOrder(arr, root->left, 2 \* i + 1, n);

root->right = insertLevelOrder(arr, root->right, 2 \* i + 2, n);

}

return root;

}

int main() {

Tree myTree;

myTree.root = NULL;

int n;

printf("Enter the number of nodes in the tree: ");

scanf("%d", &n);

int \*arr = (int \*)malloc(n \* sizeof(int));

printf("Enter the nodes in level order:\n");

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

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

}

myTree.root = insertLevelOrder(arr, myTree.root, 0, n);

printf("Tree structure:\n");

TreePrinter(myTree);

free(arr);

return 0;

}

