#include <stdio.h>

typedef struct {

float real;

float imag;

} Complex;

Complex add(Complex n1, Complex n2) {

Complex temp;

temp.real = n1.real + n2.real;

temp.imag = n1.imag + n2.imag;

return temp;

}

Complex multiply(Complex n1, Complex n2) {

Complex temp;

temp.real = n1.real \* n2.real - n1.imag \* n2.imag;

temp.imag = n1.real \* n2.imag + n1.imag \* n2.real;

return temp;

}

Complex subtract(Complex n1, Complex n2) {

Complex temp;

temp.real = n1.real - n2.real;

temp.imag = n1.imag - n2.imag;

return temp;

}

int main() {

Complex num1, num2, result;

printf("Enter the real and imaginary parts for the first complex number: ");

scanf("%f %f", &num1.real, &num1.imag);

printf("Enter the real and imaginary parts for the second complex number: ");

scanf("%f %f", &num2.real, &num2.imag);

result = add(num1, num2);

printf("Sum = %.1f + %.1fi\n", result.real, result.imag);

result = multiply(num1, num2);

printf("Product = %.1f + %.1fi\n", result.real, result.imag);

result = subtract(num1, num2);

printf("Difference = %.1f + %.1fi\n", result.real, result.imag);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

struct Student {

char name[50];

int roll;

float CGPA;

};

void readStudent(struct Student\* student) {

printf("Enter student details:\n");

printf("Name: ");

scanf("%s", student->name);

printf("Roll number: ");

scanf("%d", &student->roll);

printf("CGPA: ");

scanf("%f", &student->CGPA);

}

void displayStudent(const struct Student\* student) {

printf("Student details:\n");

printf("Name: %s\n", student->name);

printf("CGPA: %.2f\n", student->CGPA);

}

int compareRoll(const void\* a, const void\* b) {

const struct Student\* s1 = (const struct Student\*)a;

const struct Student\* s2 = (const struct Student\*)b;

return s1->roll - s2->roll;

}

int main() {

int numStudents;

printf("Enter the number of students: ");#include <stdio.h>

#include <stdlib.h>

// Define the Student structure

struct Student {

char name[50];

int roll;

float CGPA;

};

// Function to read a Student object

void readStudent(struct Student\* student) {

printf("Enter student details:\n");

printf("Name: ");

scanf("%s", student->name);

printf("Roll number: ");

scanf("%d", &student->roll);

printf("CGPA: ");

scanf("%f", &student->CGPA);

}

// Function to display a Student object

void displayStudent(const struct Student\* student) {

printf("Student details:\n");

printf("Name: %s\n", student->name);

printf("Roll number: %d\n", student->roll);

printf("CGPA: %.2f\n", student->CGPA);

}

// Comparator function for sorting based on roll number

int compareRoll(const void\* a, const void\* b) {

const struct Student\* s1 = (const struct Student\*)a;

const struct Student\* s2 = (const struct Student\*)b;

return s1->roll - s2->roll;

}

int main() {

int numStudents;

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

scanf("%d", &numStudents);

// Allocate memory for an array of Student structures

struct Student\* students = (struct Student\*)malloc(numStudents \* sizeof(struct Student));

// Read details for each student

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

readStudent(&students[i]);

}

// Display details for each student

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

displayStudent(&students[i]);

}

// Sort the array based on roll number

qsort(students, numStudents, sizeof(struct Student), compareRoll);

// Display sorted details

printf("\nSorted student details (by roll number):\n");

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

displayStudent(&students[i]);

}

// Free allocated memory

free(students);

return 0;

}

scanf("%d", &numStudents);

struct Student\* students = (struct Student\*)malloc(numStudents \* sizeof(struct Student));

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

readStudent(&students[i]);

}

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

displayStudent(&students[i]);

}

qsort(students, numStudents, sizeof(struct Student), compareRoll);

printf("\nSorted student details (by roll number):\n");

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

displayStudent(&students[i]);

}

free(students);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

struct Student {

char name[50];

int roll;

float CGPA;

};

void readStudent(struct Student\* student) {

printf("Enter student details:\n");

printf("Name: ");

scanf("%s", student->name);

printf("Roll number: ");

scanf("%d", &student->roll);

printf("CGPA: ");

scanf("%f", &student->CGPA);

}

void displayStudent(const struct Student\* student) {

printf("Student details:\n");

printf("Name: %s\n", student->name);

printf("CGPA: %.2f\n", student->CGPA);

}

int compareRoll(const void\* a, const void\* b) {

const struct Student\* s1 = (const struct Student\*)a;

const struct Student\* s2 = (const struct Student\*)b;

return s1->roll - s2->roll;

}

int main() {

int numStudents;

printf("Enter the number of students: ");#include <stdio.h>

#include <stdlib.h>

// Define the Student structure

struct Student {

char name[50];

int roll;

float CGPA;

};

// Function to read a Student object

void readStudent(struct Student\* student) {

printf("Enter student details:\n");

printf("Name: ");

scanf("%s", student->name);

printf("Roll number: ");

scanf("%d", &student->roll);

printf("CGPA: ");

scanf("%f", &student->CGPA);

}

// Function to display a Student object

void displayStudent(const struct Student\* student) {

printf("Student details:\n");

printf("Name: %s\n", student->name);

printf("Roll number: %d\n", student->roll);

printf("CGPA: %.2f\n", student->CGPA);

}

// Comparator function for sorting based on roll number

int compareRoll(const void\* a, const void\* b) {

const struct Student\* s1 = (const struct Student\*)a;

const struct Student\* s2 = (const struct Student\*)b;

return s1->roll - s2->roll;

}

int main() {

int numStudents;

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

scanf("%d", &numStudents);

// Allocate memory for an array of Student structures

struct Student\* students = (struct Student\*)malloc(numStudents \* sizeof(struct Student));

// Read details for each student

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

readStudent(&students[i]);

}

// Display details for each student

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

displayStudent(&students[i]);

}

// Sort the array based on roll number

qsort(students, numStudents, sizeof(struct Student), compareRoll);

// Display sorted details

printf("\nSorted student details (by roll number):\n");

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

displayStudent(&students[i]);

}

// Free allocated memory

free(students);

return 0;

}

scanf("%d", &numStudents);

struct Student\* students = (struct Student\*)malloc(numStudents \* sizeof(struct Student));

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

readStudent(&students[i]);

}

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

displayStudent(&students[i]);

}

qsort(students, numStudents, sizeof(struct Student), compareRoll);

printf("\nSorted student details (by roll number):\n");

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

displayStudent(&students[i]);

}

free(students);

return 0;

}

#include <stdio.h>

struct Employee

{

int id, age, salary;

char name[30], designation[30], department[30];

};

int main()

{

struct Employee e;

printf("Enter the id of the Employee: ");

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

printf("Enter the age of the Employee: ");

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

printf("Enter the name of the Employee: ");

getchar();

fgets(e.name, 30, stdin);

printf("Enter the designation of the Employee: ");

fgets(e.designation, 30, stdin);

printf("Enter the department of the Employee: ");

fgets(e.department, 30, stdin);

printf("Enter the salary of the Employee: ");

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

printf("\nEmployee Details:\n");

printf("Employee Id: %d\n", e.id);

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

printf("Employee age: %d\n", e.age);

printf("Employee designation: %s", e.designation);

printf("Employee department: %s", e.department);

printf("Employee salary: %d\n", e.salary);

return 0;

}

#include <stdio.h>

struct student {

char firstName[50];

int roll;

float marks;

} s[5];

int main() {

int i;

printf("Enter information of students:\n");

// storing information

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

s[i].roll = i + 1;

printf("\nFor roll number%d,\n", s[i].roll);

printf("Enter first name: ");

scanf("%s", s[i].firstName);

printf("Enter marks: ");

scanf("%f", &s[i].marks);

}

printf("Displaying Information:\n\n");

// displaying information

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

printf("\nRoll number: %d\n", i + 1);

printf("First name: ");

puts(s[i].firstName);

printf("Marks: %.1f", s[i].marks);

printf("\n");

}

return 0; }

#include<stdio.h>

#include<stdlib.h>

#define MAX 50

int isEmpty(int top, int stack\_arr[]);

void push(int x, int \*top, int stack\_arr[]);

int pop(int \*top, int stack\_arr[]);

void DecToBin(int num);

int main()

{

int num;

printf("Enter an integer : ");

scanf("%d",&num);

printf("Binary Equivalent is : ");

DecToBin(num);

return 0;

}

void DecToBin(int num)#include <stdio.h>

#include <stdbool.h>

#include <string.h>

bool isPalindrome(char s[]) {

int length = strlen(s);

int mid = length / 2;

int i;

char stack[mid];

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

stack[i] = s[i];

}

if (length % 2 != 0) {

i++;

}

while (s[i] != '\0') {

char ele = stack[--i];

if (ele != s[i]) {

return false;

}

i++;

}

return true;

}

int main() {

char s[] = "madam";

if (isPalindrome(s)) {

printf("Yes\n");

} else {

printf("No\n");

}

return 0;

}

{

int stack[MAX], top=-1, rem;

while(num!=0)

{

rem = num%2;

push(rem, &top, stack);

num/=2;

}

while(top!=-1)

printf("%d", pop(&top, stack));

printf("\n");

}

void push(int x, int \*top, int stack\_arr[])

{

if(\*top == (MAX-1))

printf("Stack Overflow\n");

else

{

\*top=\*top+1;

stack\_arr[\*top] = x;

}

}/\*End of push()\*/

int pop(int \*top, int stack\_arr[])

{

int x;

if(\*top == -1)

{

printf("Stack Underflow\n");

exit(1);

}

else

{

x = stack\_arr[\*top];

\*top=\*top-1;

}

return x;

}

#include<stdio.h>

int main()

{

char expression[50]; // declaration of char type array

int x=0, i=0; // declaration of two integer type variables

printf("\nEnter an expression");

scanf("%s", expression);

// Scanning the expression until we reach the end of the expression.

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

{

// Condition to check the symbol is '('

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

{

x++; // incrementing 'x' variable

}

// condition to check the symbol is ')'

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

{

x--; // decrementing 'x' variable

if(x<0)

break;

}

i++; // incrementing 'i' variable.

}

// Condition to check whether x is equal to 0 or not.

if(x==0)

{

printf("Expression is balanced");

}

else

{

printf("Expression is unbalanced");

}

return 0;

}

# define MAX 10

# define true 1

# define false 0

/\* Structure definition \*/

typedef struct

{

char item[MAX];

int top;

}stack;

void push(stack \*ps,char x);

char pop(stack \*ps);

int empty(stack \*ps);

/\* Push operation \*/

void push(stack \*ps,char x)

{

if (ps->top!=MAX-1)

{

ps->top++;

ps->item[ps->top]=x;

}

}

/\* Pop operation \*/

char pop(stack \*ps)

{

if(!empty(ps))

return(ps->item[ps->top--]);

}

/\* Stack empty operation \*/

int empty(stack \*ps)

{

if (ps->top==-1)

return(true);

else

return(false);

}