51. Write macro definitions for the following: 1. To test whether a character is a lowercase letter or not. 2. To test whether a character is an uppercase letter or not. 3. To test whether a character is an alphabet or not. Make use of the macros you defined in 1 and 2 above. 4. To obtain the bigger of two numbers.

# include

#define ISUPPER(x) ( x >= 65 && x <= 90 ? 1 : 0 )

#define ISLOWER(x) ( x >= 97 && x <= 122 ? 1 : 0 )

#define ISALPHA(x) ( ISUPPER(x) || ISLOWER(x) )

#define BIG(x,y) ( x > y ? x : y )

int main( )

{

char ch ; int d, a, b ;

printf ( "\nEnter any alphabet/character: " ) ;

scanf ( "%c", &ch ) ;

if ( ISUPPER ( ch ) == 1 )

printf ( "You entered a capital letter\n" ) ;

if ( ISLOWER ( ch ) == 1 )

printf ( "You entered a small case letter\n" ) ;

if ( ISALPHA ( ch ) != 1 )

printf ( "You entered character other than an alphabet\n" ) ;

printf ( "Enter any two numbers: " ) ;

scanf ( "%d%d", &a, &b ) ;

d = BIG ( a, b ) ;

printf ( "Bigger number is %d\n", d ) ;

return 0 ;

}

52. Write a program that interchanges elements at odd position with elements at even position in an array of 10 elements.

int main( )

{

int num[ ] = { 12, 4, 5, 1, 9, 13, 11, 19, 54, 34 } ;

int i, t ;

for ( i = 0 ; i <= 9 ; i = i + 2 )

{

t = num[ i ] ; num [ i ] = num [ i + 1 ] ; num [ i + 1 ] = t ;

}

for ( i = 0 ; i <= 9 ; i++ )

printf ( "%d\t", num[ i ] ) ;

return 0 ;

}

53. Write a program to copy the contents of a 5-element integer array into another array in reverse order.

int arr1[ 5 ], arr2[ 5 ], i, j ;

printf ( "\nEnter 5 elements of array:\n" ) ;

for ( i = 0 ; i <= 4 ; i++ )

scanf ( "%d", &arr1[ i ] ) ;

for ( i = 0, j = 4 ; i <= 4 ; i++, j-- ) arr2[ j ] = arr1[ i ] ;

printf ( "Elements in reverse order:\n" ) ;

for ( i = 0 ; i <= 4 ; i++ )

printf ( "%d\t", arr2[ i ] ) ;

return 0 ;

54. An array contains 10 integers. Receive the number to be searched in the array as input. Write a program to search this number in the array and display the number of times it occurs in the array.

int num[ ] = { 7, 3, 5, 4, 6, 7, 2, 4, 6, 7 } ;

int n, i, count ;

printf ( "\nEnter an element to search: " ) ;

scanf ( "%d", &n ) ;

count = 0 ; for ( i = 0 ; i <= 9 ; i++ )

{

if ( num[ i ] == n ) count++ ;

}

printf ( "Number %d is found %d time(s) in the array\n", n, count ) ;

return 0 ;

55. Twenty-five numbers are entered from the keyboard into an array. Write a program to find out how many of them are positive, how many are negative, how many are even and how many odd.

#include <stdio.h>

int main() {

int numbers[25];

int positive\_count = 0, negative\_count = 0, even\_count = 0, odd\_count = 0;

printf("Enter 25 numbers:\n");

for (int i = 0; i < 25; i++) {

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

}

for (int i = 0; i < 25; i++) {

if (numbers[i] > 0)

positive\_count++;

else if (numbers[i] < 0)

negative\_count++;

if (numbers[i] % 2 == 0)

even\_count++;

else

odd\_count++;

}

printf("Positive numbers: %d\n", positive\_count);

printf("Negative numbers: %d\n", negative\_count);

printf("Even numbers: %d\n", even\_count);

printf("Odd numbers: %d\n", odd\_count);

return 0;

}

56. If an array arr contains n elements, then write a program to check if arr[ 0 ] = arr[ n - 1 ], arr[ 1 ] = arr[ n - 2 ] and so on.

#include <stdio.h>

int main() {

int n;

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

scanf("%d", &n);

int arr[n];

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

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

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

}

int flag = 1;

for (int i = 0; i < n / 2; i++) {

if (arr[i] != arr[n - 1 - i]) {

flag = 0;

break;

}

}

if (flag)

printf("The array satisfies the condition.\n");

else

printf("The array does not satisfy the condition.\n");

return 0;

}

57. Write a program using pointers to find the smallest number in an array of 25 integers.

#include <stdio.h>

int main() {

int numbers[25];

printf("Enter 25 integers:\n");

for (int i = 0; i < 25; i++) {

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

}

int \*ptr = numbers;

int smallest = \*ptr;

for (int i = 1; i < 25; i++) {

if (\*(ptr + i) < smallest) {

smallest = \*(ptr + i);

}

}

printf("The smallest number is: %d\n", smallest);

return 0;

}

58. Write a program to pick up the largest number from a 5 row by 5 column matrix.

#include <stdio.h>

int main() {

int matrix[5][5];

printf("Enter the elements of the 5x5 matrix:\n");

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

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

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

}

}

int largest = matrix[0][0];

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

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

if (matrix[i][j] > largest) {

largest = matrix[i][j];

}

}

}

printf("The largest number in the matrix is: %d\n", largest);

return 0;

}

59. Write a program to obtain transpose of a 3 x 5 matrix. The transpose of a matrix is obtained by exchanging the elements of each row with the elements of the corresponding column

#include <stdio.h>

int main() {

int matrix[3][5];

int transpose[5][3];

printf("Enter the elements of the 3x5 matrix:\n");

for (int i = 0; i < 3; i++) {

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

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

}

}

// Transpose the matrix

for (int i = 0; i < 3; i++) {

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

transpose[j][i] = matrix[i][j];

}

}

printf("The transpose of the matrix is:\n");

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

for (int j = 0; j < 3; j++) {

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

}

printf("\n");

}

return 0;

}

60. Write a program to find if a square matrix is symmetric.

#include <stdio.h>

#define SIZE 3

int main() {

int matrix[SIZE][SIZE];

printf("Enter the elements of the %dx%d matrix:\n", SIZE, SIZE);

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

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

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

}

}

int symmetric = 1;

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

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

if (matrix[i][j] != matrix[j][i]) {

symmetric = 0;

break;

}

}

if (!symmetric) {

break;

}

}

if (symmetric) {

printf("The matrix is symmetric.\n");

} else {

printf("The matrix is not symmetric.\n");

}

return 0;

}

61. Write a program to add two 6 x 6 matrices.

#include <stdio.h>

#define ROWS 6

#define COLS 6

int main() {

int matrix1[ROWS][COLS], matrix2[ROWS][COLS], sum[ROWS][COLS];

printf("Enter elements of the first 6x6 matrix:\n");

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

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

scanf("%d", &matrix1[i][j]);

}

}

printf("Enter elements of the second 6x6 matrix:\n");

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

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

scanf("%d", &matrix2[i][j]);

}

}

printf("Sum of the matrices:\n");

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

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

sum[i][j] = matrix1[i][j] + matrix2[i][j];

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

}

printf("\n");

}

return 0;

}

62. Write a program to multiply any two 3 x 3 matrices

#include <stdio.h>

#define SIZE 3

void multiplyMatrices(int mat1[][SIZE], int mat2[][SIZE], int result[][SIZE]) {

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

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

result[i][j] = 0;

for (int k = 0; k < SIZE; k++) {

result[i][j] += mat1[i][k] \* mat2[k][j];

}

}

}

}

void displayMatrix(int mat[][SIZE]) {

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

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

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

}

printf("\n");

}

}

int main() {

int mat1[SIZE][SIZE], mat2[SIZE][SIZE], result[SIZE][SIZE];

printf("Enter elements of the first 3x3 matrix:\n");

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

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

scanf("%d", &mat1[i][j]);

}

}

printf("Enter elements of the second 3x3 matrix:\n");

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

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

scanf("%d", &mat2[i][j]);

}

}

multiplyMatrices(mat1, mat2, result);

printf("Result of matrix multiplication:\n");

displayMatrix(result);

return 0;

}

63. Given an array p[ 5 ], write a function to shift it circularly left by two positions. Thus, if the original array is { 15, 30, 28, 19, 61 } then after shifting it will be { 28, 19, 61, 15, 30 } Call this function for a 4 x 5 matrix and get its rows left shifted.

#include <stdio.h>

#define ROWS 4

#define COLS 5

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

int temp = arr[0];

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

arr[i] = arr[i + 1];

}

arr[n - 1] = temp;

}

void displayMatrix(int mat[][COLS], int rows) {

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

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

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

}

printf("\n");

}

}

void shiftRowsLeft(int mat[][COLS], int rows) {

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

circularLeftShift(mat[i], COLS);

}

}

int main() {

int matrix[ROWS][COLS] = {

{15, 30, 28, 19, 61},

{7, 8, 9, 10, 11},

{12, 13, 14, 15, 16},

{17, 18, 19, 20, 21}

};

printf("Original matrix:\n");

displayMatrix(matrix, ROWS);

shiftRowsLeft(matrix, ROWS);

printf("\nMatrix after circular left shift of rows:\n");

displayMatrix(matrix, ROWS);

return 0;

}

64. Write a program that extracts part of the given string from the specified position. For example, if from the sting "Working with strings is fun", starting from position 3, 4 characters are extracted then it should return "king"

#include <stdio.h>

#include <string.h>

void extractSubstring(char source[], int start, int length, char result[]) {

int index = 0;

for (int i = start - 1; i < start - 1 + length; i++) {

result[index++] = source[i];

}

result[index] = '\0';

}

int main() {

char source[] = "Working with strings is fun";

int start = 3;

int length = 4;

char result[length + 1];

extractSubstring(source, start, length, result);

printf("Extracted substring: %s\n", result);

return 0;

}

65. Write a program that converts a string like "124" to an integer 124.

#include <stdio.h>

int stringToInt(char str[]) {

int result = 0;

int i = 0;

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

result = result \* 10 + (str[i] - '0');

i++;

}

return result;

}

int main() {

char str[] = "124";

int num = stringToInt(str);

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

return 0;

}

66. Write a program that generates and prints the Fibonacci words of order 0 through 5. For example, f(0) = "A", f(1) = "B", f(2) = "BA", f(3) = "BAB", f(4) = "BABBA", etc.

#include <stdio.h>

#include <string.h>

void generateFibonacciWords(int n) {

char fibWords[n + 2][100];

strcpy(fibWords[0], "A");

strcpy(fibWords[1], "B");

for (int i = 2; i <= n; i++) {

strcpy(fibWords[i], fibWords[i - 1]);

strcat(fibWords[i], fibWords[i - 2]);

}

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

printf("f(%d) = %s\n", i, fibWords[i]);

}

}

int main() {

printf("Fibonacci words of order 0 through 5:\n");

generateFibonacciWords(5);

return 0;

}

67. Write a program to store a few strings using an array of pointers to strings. Receive a string and check if it is present in the array.

int main() {

char \*strings[MAX\_STRINGS] = {"apple", "banana", "orange", "grape", "kiwi"};

char input[MAX\_LENGTH];

printf("Enter a string to check: ");

scanf("%s", input);

int found = 0;

for (int i = 0; i < MAX\_STRINGS; i++) {

if (strcmp(strings[i], input) == 0) {

found = 1;

break;

}

}

if (found) {

printf("String '%s' is present in the array.\n", input);

} else {

printf("String '%s' is not present in the array.\n", input);

}

return 0;

}

68. Write a program to alphabetically sort a set of names stored using an array of pointers to strings.

void sortNames(char \*names[], int n) {

char \*temp;

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

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

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

temp = names[i];

names[i] = names[j];

names[j] = temp;

}

}

}

}

int main() {

char \*names[MAX\_NAMES] = {"John", "Alice", "Michael", "Emma", "David"};

printf("Names before sorting:\n");

for (int i = 0; i < MAX\_NAMES; i++) {

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

}

sortNames(names, MAX\_NAMES);

printf("\nNames after sorting:\n");

for (int i = 0; i < MAX\_NAMES; i++) {

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

}

return 0;

}

69. Write a program to reverse the strings stored in an array of pointers to strings:

void reverseStrings(char \*strings[], int n) {

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

int len = strlen(strings[i]);

for (int j = 0; j < len / 2; j++) {

char temp = strings[i][j];

strings[i][j] = strings[i][len - 1 - j];

strings[i][len - 1 - j] = temp;

}

}

}

int main() {

char \*strings[MAX\_STRINGS] = {"apple", "banana", "orange", "grape", "kiwi"};

printf("Strings before reversal:\n");

for (int i = 0; i < MAX\_STRINGS; i++) {

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

}

reverseStrings(strings, MAX\_STRINGS);

printf("\nStrings after reversal:\n");

for (int i = 0; i < MAX\_STRINGS; i++) {

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

}

return 0;

}

70. Write a program to count the number of occurrences of any two vowels in succession in a line of text. For example, in the following sentence:

“Please read this application and give me gratuity”

such occurrences are ea, ea, ui.

int countSuccessiveVowels(char \*text) {

int count = 0;

int length = strlen(text);

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

if ((text[i] == 'a' || text[i] == 'e' || text[i] == 'i' || text[i] == 'o' || text[i] == 'u' ||

text[i] == 'A' || text[i] == 'E' || text[i] == 'I' || text[i] == 'O' || text[i] == 'U') &&

(text[i + 1] == 'a' || text[i + 1] == 'e' || text[i + 1] == 'i' || text[i + 1] == 'o' ||

text[i + 1] == 'u' || text[i + 1] == 'A' || text[i + 1] == 'E' || text[i + 1] == 'I' ||

text[i + 1] == 'O' || text[i + 1] == 'U')) {

count++;

printf("%c%c\n", text[i], text[i + 1]);

}

}

return count;

}

int main() {

char text[100];

printf("Enter a line of text: ");

fgets(text, sizeof(text), stdin);

int occurrences = countSuccessiveVowels(text);

printf("Number of occurrences of any two vowels in succession: %d\n", occurrences);

return 0;

}

71. Write a program that receives an integer (less than or equal to nine digits in length) and prints out the number in words. For example, if the number input is 12342, then the output should be Twelve Thousand Three Hundred Forty Two.

void convertToWords(int num) {

char \*ones[] = {"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};

char \*teens[] = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};

char \*tens[] = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};

if (num == 0) {

printf("Zero");

return;

}

if (num >= 1000000) {

convertToWords(num / 1000000);

printf(" Million ");

num %= 1000000;

}

if (num >= 1000) {

convertToWords(num / 1000);

printf(" Thousand ");

num %= 1000;

}

if (num >= 100) {

convertToWords(num / 100);

printf(" Hundred ");

num %= 100;

}

if (num >= 20) {

printf("%s ", tens[num / 10]);

num %= 10;

}

if (num >= 10) {

printf("%s ", teens[num - 10]);

return;

}

if (num > 0) {

printf("%s ", ones[num]);

}

}

int main() {

int num;

printf("Enter an integer less than or equal to nine digits: ");

scanf("%d", &num);

printf("In words: ");

convertToWords(num);

return 0;

}

72. A stack is a data structure in which addition of new element or deletion of existing element always takes place at the same end known as ‘top’ of stack. Write a program to implement a stack using a linked list.

struct Node {

int data;

struct Node\* next;

};

struct Node\* top = NULL;

void push(int value) {

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

if (newNode == NULL) {

printf("Stack overflow\n");

return;

}

newNode->data = value;

newNode->next = top;

top = newNode;

}

void pop() {

if (top == NULL) {

printf("Stack underflow\n");

return;

}

struct Node\* temp = top;

top = top->next;

free(temp);

}

int peek() {

if (top == NULL) {

printf("Stack is empty\n");

return -1;

}

return top->data;

}

void display() {

if (top == NULL) {

printf("Stack is empty\n");

return;

}

struct Node\* temp = top;

printf("Stack elements: ");

while (temp != NULL) {

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

temp = temp->next;

}

printf("\n");

}

73. In a data structure called queue the addition of new element takes place at the end (called ‘rear’ of queue), whereas deletion takes place at the other end (called ‘front’ of queue). Write a program to implement a queue using a linked list.

struct Node {

int data;

struct Node\* next;

};

struct Node\* front = NULL;

struct Node\* rear = NULL;

void enqueue(int value) {

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

if (newNode == NULL) {

printf("Queue overflow\n");

return;

}

newNode->data = value;

newNode->next = NULL;

if (front == NULL) {

front = rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

}

void dequeue() {

if (front == NULL) {

printf("Queue underflow\n");

return;

}

struct Node\* temp = front;

front = front->next;

if (front == NULL) {

rear = NULL;

}

free(temp);

}

void display() {

if (front == NULL) {

printf("Queue is empty\n");

return;

}

struct Node\* temp = front;

printf("Queue elements: ");

while (temp != NULL) {

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

temp = temp->next;

}

printf("\n");

}

74. Create a structure called student that can contain data given below: Roll number, Name, Department, Course, Year of joining Assume that there are not more than 450 students in the college. (1) Write a function to print names of all students who joined in a particular year. (2) Write a function to print the data of a student whose roll number is received by the function.

#define MAX\_STUDENTS 450

#define MAX\_NAME\_LENGTH 50

struct Student {

int rollNumber;

char name[MAX\_NAME\_LENGTH];

char department[MAX\_NAME\_LENGTH];

char course[MAX\_NAME\_LENGTH];

int yearOfJoining;

};

struct Student students[MAX\_STUDENTS];

int studentCount = 0;

void printStudentsByYear(int year) {

printf("Students who joined in year %d:\n", year);

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

if (students[i].yearOfJoining == year) {

printf("Roll Number: %d, Name: %s, Department: %s, Course: %s\n",

students[i].rollNumber, students[i].name, students[i].department, students[i].course);

}

}

}

void printStudentByRollNumber(int rollNumber) {

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

if (students[i].rollNumber == rollNumber) {

printf("Student Details:\n");

printf("Roll Number: %d\n", students[i].rollNumber);

printf("Name: %s\n", students[i].name);

printf("Department: %s\n", students[i].department);

printf("Course: %s\n", students[i].course);

printf("Year of Joining: %d\n", students[i].yearOfJoining);

return;

}

}

printf("Student with roll number %d not found.\n", rollNumber);

}

75. Suppose there is a structure called employee that holds information like employee code, name and date of joining. Write a program to create an array of structures and enter some data into it. Then ask the user to enter current date. Display the names of those employees whose tenure is greater than equal to 3 years.

struct Employee {

int employeeCode;

char name[MAX\_NAME\_LENGTH];

int joiningYear;

};

struct Employee employees[MAX\_EMPLOYEES];

int employeeCount = 0;

void displayEmployeesWithTenure(int currentYear) {

printf("Employees with tenure >= 3 years:\n");

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

int tenure = currentYear - employees[i].joiningYear;

if (tenure >= 3) {

printf("%s\n", employees[i].name);

}

}

}

76. Write a program to read a file and display its contents along with line numbers before each line

int main( )

{

FILE \*fp ;

char ch, source[ 67 ] ;

int count = 1 ;

printf ( "\nEnter file name: " ) ;

scanf ( "%s", source ) ;

fp = fopen ( source, "r" ) ;

if ( fp == NULL ) { puts ( "Unable to open the file." ) ;

exit ( 0 ) ;

}

printf ( "\n%3d: ", count ) ;

while ( ( ch = getc( fp ) ) != EOF )

{

if ( ch == '\n' )

{

count++ ;

printf ( "\n%3d: ", count ) ;

}

else printf ( "%c", ch ) ;

}

fclose ( fp ) ;

return 0 ;

}

77. Write a program to append the contents of one file at the end of another.

int main() {

FILE \*sourceFile, \*destinationFile;

char sourceFileName[100], destinationFileName[100];

char ch;

printf("Enter the source file name: ");

scanf("%s", sourceFileName);

printf("Enter the destination file name: ");

scanf("%s", destinationFileName);

sourceFile = fopen(sourceFileName, "r");

if (sourceFile == NULL) {

printf("Unable to open source file '%s'\n", sourceFileName);

exit(1);

}

destinationFile = fopen(destinationFileName, "a");

if (destinationFile == NULL) {

printf("Unable to open destination file '%s'\n", destinationFileName);

fclose(sourceFile);

exit(1);

}

while ((ch = fgetc(sourceFile)) != EOF) {

fputc(ch, destinationFile);

}

printf("Contents of '%s' appended to '%s' successfully.\n", sourceFileName, destinationFileName);

fclose(sourceFile);

fclose(destinationFile);

return 0;

}

78. Suppose a file contains student records with each record containing name and age of a student. Write a program to read these records and display them in sorted order by name

struct Student {

char name[MAX\_NAME\_LENGTH];

int age;

};

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

return strcmp(((struct Student\*)a)->name, ((struct Student\*)b)->name);

}

int main() {

FILE \*file;

char fileName[100];

struct Student students[MAX\_STUDENTS];

int studentCount = 0;

printf("Enter the file name: ");

scanf("%s", fileName);

file = fopen(fileName, "r");

if (file == NULL) {

printf("Unable to open file '%s'\n", fileName);

exit(1);

}

while (fscanf(file, "%s %d", students[studentCount].name, &students[studentCount].age) != EOF) {

studentCount++;

}

fclose(file);

qsort(students, studentCount, sizeof(struct Student), compareStudents);

printf("Sorted student records by name:\n");

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

printf("%s %d\n", students[i].name, students[i].age);

}

return 0;

}

79. Write a program to copy contents of one file to another. While doing so replace all lowercase characters to their equivalent uppercase characters.

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

int main() {

FILE \*sourceFile, \*destinationFile;

char sourceFileName[100], destinationFileName[100];

char ch;

printf("Enter the source file name: ");

scanf("%s", sourceFileName);

printf("Enter the destination file name: ");

scanf("%s", destinationFileName);

sourceFile = fopen(sourceFileName, "r");

if (sourceFile == NULL) {

printf("Unable to open source file '%s'\n", sourceFileName);

exit(1);

}

destinationFile = fopen(destinationFileName, "w");

if (destinationFile == NULL) {

printf("Unable to open destination file '%s'\n", destinationFileName);

fclose(sourceFile);

exit(1);

}

while ((ch = fgetc(sourceFile)) != EOF) {

if (islower(ch)) {

ch = toupper(ch);

}

fputc(ch, destinationFile);

}

printf("Contents of '%s' copied to '%s' with lowercase characters replaced by uppercase characters.\n", sourceFileName, destinationFileName);

fclose(sourceFile);

fclose(destinationFile);

return 0;

}

80. Write a program that merges lines alternately from two files and writes the results to a new file. If one file has a smaller number of lines than the other, the remaining lines from the larger file should be simply copied into the target file.

int main() {

FILE \*file1, \*file2, \*targetFile;

char file1Name[100], file2Name[100], targetFileName[100];

char line[1000];

printf("Enter the first file name: ");

scanf("%s", file1Name);

printf("Enter the second file name: ");

scanf("%s", file2Name);

printf("Enter the target file name: ");

scanf("%s", targetFileName);

file1 = fopen(file1Name, "r");

if (file1 == NULL) {

printf("Unable to open file '%s'\n", file1Name);

exit(1);

}

file2 = fopen(file2Name, "r");

if (file2 == NULL) {

printf("Unable to open file '%s'\n", file2Name);

fclose(file1);

exit(1);

}

targetFile = fopen(targetFileName, "w");

if (targetFile == NULL) {

printf("Unable to create file '%s'\n", targetFileName);

fclose(file1);

fclose(file2);

exit(1);

}

while (fgets(line, sizeof(line), file1) != NULL || fgets(line, sizeof(line), file2) != NULL) {

if (line[0] != '\0') {

fputs(line, targetFile);

}

}

fclose(file1);

fclose(file2);

fclose(targetFile);

printf("Merged lines from '%s' and '%s' written to '%s'.\n", file1Name, file2Name, targetFileName);

return 0;

}

81. Given a text file, write a program to create another text file deleting the words “a”, “the”, “an” and replacing each one of them with a blank space.

int main() {

FILE \*sourceFile, \*targetFile;

char sourceFileName[100], targetFileName[100];

char word[MAX\_WORD\_LENGTH];

printf("Enter the source file name: ");

scanf("%s", sourceFileName);

printf("Enter the target file name: ");

scanf("%s", targetFileName);

sourceFile = fopen(sourceFileName, "r");

if (sourceFile == NULL) {

printf("Unable to open source file '%s'\n", sourceFileName);

exit(1);

}

targetFile = fopen(targetFileName, "w");

if (targetFile == NULL) {

printf("Unable to create file '%s'\n", targetFileName);

fclose(sourceFile);

exit(1);

}

while (fscanf(sourceFile, "%s", word) != EOF) {

if (strcmp(word, "a") == 0 || strcmp(word, "the") == 0 || strcmp(word, "an") == 0) {

fprintf(targetFile, " ");

} else {

fprintf(targetFile, "%s ", word);

}

}

fclose(sourceFile);

fclose(targetFile);

printf("Words 'a', 'the', 'an' replaced with blank space in '%s' and written to '%s'.\n", sourceFileName, targetFileName);

return 0;

}

82. Write a program using command-line arguments to search for a word in a file and replace it with the specified word. The usage of the program is shown below. C> change

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

if (argc != 4) {

printf("Usage: %s <old word> <new word> <filename>\n", argv[0]);

exit(1);

}

char \*oldWord = argv[1];

char \*newWord = argv[2];

char \*fileName = argv[3];

FILE \*file = fopen(fileName, "r");

if (file == NULL) {

printf("Unable to open file '%s'\n", fileName);

exit(1);

}

FILE \*tempFile = fopen("temp.txt", "w");

if (tempFile == NULL) {

printf("Unable to create temporary file\n");

fclose(file);

exit(1);

}

char line[MAX\_LINE\_LENGTH];

while (fgets(line, sizeof(line), file) != NULL) {

char \*pos;

while ((pos = strstr(line, oldWord)) != NULL) {

fputs(line, tempFile);

int index = pos - line;

fputs(newWord, tempFile);

line[index] = ' ';

strncpy(line + index + strlen(newWord), pos + strlen(oldWord), strlen(pos + strlen(oldWord)) + 1);

}

fputs(line, tempFile);

}

fclose(file);

fclose(tempFile);

remove(fileName);

rename("temp.txt", fileName);

printf("'%s' replaced with '%s' in file '%s'\n", oldWord, newWord, fileName);

return 0;

}

83. Write a program that can be used at command prompt as a calculating utility. The usage of the program is shown below. C> calc where, n and m are two integer operands and switch is either an arithmetic operator or a comparison operator. If arithmetic operator is supplied, the output should be the result of the operation. If comparison operator is supplied then the output should be True or False.

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

if (argc != 4) {

printf("Usage: %s <switch> <n> <m>\n", argv[0]);

exit(1);

}

char op = argv[1][0];

int n = atoi(argv[2]);

int m = atoi(argv[3]);

bool result;

switch (op) {

case '+':

printf("%d\n", n + m);

break;

case '-':

printf("%d\n", n - m);

break;

case '\*':

printf("%d\n", n \* m);

break;

case '/':

if (m == 0) {

printf("Error: Division by zero\n");

exit(1);

}

printf("%d\n", n / m);

break;

case '>':

result = (n > m);

printf("%s\n", result ? "True" : "False");

break;

case '<':

result = (n < m);

printf("%s\n", result ? "True" : "False");

break;

case '=':

result = (n == m);

printf("%s\n", result ? "True" : "False");

break;

default:

printf("Invalid switch '%c'\n", op);

exit(1);

}

return 0;

}

84. The information about colors is to be stored in bits of an unsigned char variable called color. Bit numbers 0 to 6, each represent 7 colors of a rainbow, i.e., bit 0 represents violet, 1 represents indigo, and so on. Write a program that asks the user to enter a number and based on this number it reports which colors in the rainbow do the number represents.

#include <stdio.h>

int main() {

unsigned char color;

int num;

printf("Enter a number: ");

scanf("%d", &num);

color = (unsigned char)num;

printf("The number %d represents the following colors in the rainbow:\n", num);

if (color & (1 << 0)) printf("Violet\n");

if (color & (1 << 1)) printf("Indigo\n");

if (color & (1 << 2)) printf("Blue\n");

if (color & (1 << 3)) printf("Green\n");

if (color & (1 << 4)) printf("Yellow\n");

if (color & (1 << 5)) printf("Orange\n");

if (color & (1 << 6)) printf("Red\n");

return 0;

}

85. Write a program to receive an unsigned 16-bit integer and then exchange the contents of its 2 bytes using bitwise operators.

#include <stdio.h>

int main() {

unsigned short int num, swapped;

printf("Enter an unsigned 16-bit integer: ");

scanf("%hu", &num);

swapped = ((num & 0xFF) << 8) | ((num >> 8) & 0xFF);

printf("Original number: %hu\n", num);

printf("Number after exchanging bytes: %hu\n", swapped);

return 0;

}

86. Write a program to receive an 8-bit number into a variable and then exchange its higher 4 bits with lower 4 bits

#include <stdio.h>

int main() {

unsigned char num, swapped;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

swapped = ((num & 0x0F) << 4) | ((num & 0xF0) >> 4);

printf("Original number: %hhu\n", num);

printf("Number after exchanging higher 4 bits with lower 4 bits: %hhu\n", swapped);

return 0;

}

87. Write a program to receive an 8-bit number into a variable and then set its odd bits to 1.

#include <stdio.h>

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

num |= 0xAA;

printf("Number after setting odd bits to 1: %hhu\n", num);

return 0;

}

88. Write a program to receive an 8-bit number into a variable and then check if its 3rd and 5th bit are on. If these bits are found to be on then put them off.

#include <stdio.h>

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

if ((num & 0x20) && (num & 0x08)) {

num &= ~(0x20);

num &= ~(0x08);

printf("3rd and 5th bits were on. They are now off.\n");

} else {

printf("3rd and/or 5th bits were not on.\n");

}

printf("Number after checking and turning off bits: %hhu\n", num);

return 0;

}

89. Write a program to receive an 8-bit number into a variable and then check if its 3rd and 5th bit are off. If these bits are found to be off then put them on.

#include <stdio.h>

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

if (!(num & 0x20) && !(num & 0x08)) {

num |= 0x20;

num |= 0x08;

printf("3rd and 5th bits were off. They are now on.\n");

} else {

printf("3rd and/or 5th bits were not off.\n");

}

printf("Number after checking and turning on bits: %hhu\n", num);

return 0;

}

90. Define three functions—fun1( ), fun2( ) and fun3( ). Each function should receive two integers and return a float. Store the addresses of these functions in an array. Call these functions using the addresses stored in the array.

#include <stdio.h>

float fun1(int a, int b) {

return (float)(a + b);

}

float fun2(int a, int b) {

return (float)(a - b);

}

float fun3(int a, int b) {

return (float)(a \* b);

}

int main() {

float (\*funcPointers[3])(int, int) = {fun1, fun2, fun3};

int a = 10, b = 5;

for (int i = 0; i < 3; i++) {

printf("Result of function %d: %.2f\n", i + 1, funcPointers[i](a, b));

}

return 0;

}

91. Define a function which can find average of the arguments passed to it. Note that in different calls the function may receive different number of arguments.

#include <stdio.h>

#include <stdarg.h>

double find\_average(int count, ...) {

va\_list args;

double sum = 0.0;

va\_start(args, count);

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

sum += va\_arg(args, int);

}

va\_end(args);

return count ? sum / count : 0;

}

int main() {

printf("Average of 3, 5, 7: %.2f\n", find\_average(3, 3, 5, 7));

printf("Average of 10, 20: %.2f\n", find\_average(2, 10, 20));

printf("Average of 1, 2, 3, 4, 5: %.2f\n", find\_average(5, 1, 2, 3, 4, 5));

return 0;

}

92. Write a program, which stores information about a date in a structure containing three members—day, month and year. Using bit fields, the day number should get stored in first 5 bits of day, the month number in 4 bits of month and year in 12 bits of year. Write a program to read date of joining of 10 employees and display them in ascending order of year.

#include <stdio.h>

#define EMP\_COUNT 10

typedef struct {

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 12;

} Date;

void sort\_dates(Date dates[], int n) {

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

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

if (dates[j].year > dates[j + 1].year) {

Date temp = dates[j];

dates[j] = dates[j + 1];

dates[j + 1] = temp;

}

}

}

}

int main() {

Date dates[EMP\_COUNT];

for (int i = 0; i < EMP\_COUNT; i++) {

printf("Enter joining date of employee %d (dd mm yyyy): ", i + 1);

scanf("%u %u %u", &dates[i].day, &dates[i].month, &dates[i].year);

}

sort\_dates(dates, EMP\_COUNT);

printf("\nDates of joining in ascending order of year:\n");

for (int i = 0; i < EMP\_COUNT; i++) {

printf("%02u-%02u-%04u\n", dates[i].day, dates[i].month, dates[i].year);

}

return 0;

}

93. Write a program to read and store information about insurance policy holder. The information contains details like gender, whether the holder is minor/major, policy name and duration of the policy. Make use of bit-fields to store this information.

#include <stdio.h>

#include <string.h>

#define POLICY\_NAME\_LEN 50

#define MAX\_HOLDERS 10

typedef struct {

unsigned int gender : 1;

unsigned int is\_minor : 1;

char policy\_name[POLICY\_NAME\_LEN];

unsigned int duration : 6;

} PolicyHolder;

int main() {

PolicyHolder holders[MAX\_HOLDERS];

int n;

printf("Enter the number of policy holders: ");

scanf("%d", &n);

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

int gender, is\_minor;

printf("Enter details for policy holder %d\n", i + 1);

printf("Gender (0 for female, 1 for male): ");

scanf("%d", &gender);

printf("Is Minor (0 for major, 1 for minor): ");

scanf("%d", &is\_minor);

printf("Policy Name: ");

scanf(" %[^\n]%\*c", holders[i].policy\_name);

printf("Duration of Policy (in years): ");

scanf("%u", &holders[i].duration);

holders[i].gender = gender;

holders[i].is\_minor = is\_minor;

}

printf("\nPolicy Holder Information:\n");

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

printf("Policy Holder %d:\n", i + 1);

printf("Gender: %s\n", holders[i].gender ? "Male" : "Female");

printf("Minor/Major: %s\n", holders[i].is\_minor ? "Minor" : "Major");

printf("Policy Name: %s\n", holders[i].policy\_name);

printf("Duration: %u years\n", holders[i].duration);

}

return 0;

}

94. Shifting

#include <stdio.h>

void showbits(unsigned char n);

int main() {

char num = 5, j, k;

printf("\nDecimal %d is same as binary ", num);

showbits(num);

for (j = 1; j <= 3; j++) {

k = num << j;

printf("\n%d left shift %d gives ", num, j);

showbits(k);

}

return 0;

}

void showbits(unsigned char n) {

int i;

unsigned char j, k, andmask;

for (i = 7; i >= 0; i--) {

j = i;

andmask = 1 << j;

k = n & andmask;

k == 0 ? printf("0") : printf("1");

}

}

Find the errors

94. # include <stdio.h>

int main( )

{

int i ;

char a[ ] = "Hello" ;

while ( a != '\0' )

{

printf ( "%c", \*a ) ;

a++ ;

}

return 0 ;

}

->#include <stdio.h>

int main() {

int i;

char a[] = "Hello";

char \*p = a;

while (\*p != '\0') {

printf("%c", \*p);

p++;

}

return 0;

}

95. # include<stido.h>

int main( )

{

double dval ;

scanf ( "%f", &dval ) ;

printf ( "Double Value = %lf\n", dval ) ;

return 0 ;

}

->#include <stdio.h>

int main() {

double dval;

scanf("%lf", &dval);

printf("Double Value = %lf\n", dval);

return 0;

}

96. # include<stdiio.h>

int main( )

{

int ival ;

scanf ( "%d\n", &n ) ;

printf ( "Integer Value = %d\n", ival ) ;

return 0 ;

}

->#include <stdio.h>

int main() {

int ival;

scanf("%d", &ival);

printf("Integer Value = %d\n", ival);

return 0;

}

97. # include<stdio.h>

int main( )

{

int dd, mm, yy ;

printf ( "Enter date in dd/mm/yy or dd-mm-yy format\n" ) ;

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

printf ( "The date is: %d - %d - %d\n", dd, mm, yy ) ;

return 0 ;

}

->#include <stdio.h>

int main() {

int dd, mm, yy;

printf("Enter date in dd/mm/yy or dd-mm-yy format\n");

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

printf("The date is: %d - %d - %d\n", dd, mm, yy);

return 0;

}

98. # include <stdio.h>

int main( )

{

char text ;

sprintf ( text, "%4d\t%2.2f\n%s", 12, 3.452, "Merry Go Round" ) ;

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

return 0 ;

}

->#include <stdio.h>

int main() {

char text[100];

sprintf(text, "%4d\t%2.2f\n%s", 12, 3.452, "Merry Go Round");

printf("%s", text);

return 0;

}

99. An automobile company has serial number for engine parts starting from AA0 to FF9. The other characteristics of parts are year of manufacture, material and quantity manufactured. (1) Create a structure to store information corresponding to a part. (2) Write a program to retrieve information on parts with serial numbers between BB1 and CC6.

#include <stdio.h>

#include <string.h>

#define SERIAL\_LENGTH 4

struct Part {

char serial[SERIAL\_LENGTH];

int year;

char material[20];

int quantity;

};

int main() {

struct Part parts[] = {

{"AA0", 2020, "Steel", 100},

{"BB1", 2021, "Aluminum", 150},

{"CC2", 2019, "Plastic", 200},

{"DD3", 2022, "Carbon Fiber", 120},

{"EE4", 2018, "Steel", 180},

{"FF5", 2023, "Aluminum", 220},

{"GG6", 2020, "Plastic", 300},

{"HH7", 2021, "Carbon Fiber", 250},

{"II8", 2019, "Steel", 170},

{"JJ9", 2022, "Aluminum", 280}

};

printf("Parts between BB1 and CC6:\n");

for (int i = 0; i < sizeof(parts) / sizeof(parts[0]); ++i) {

if (strcmp(parts[i].serial, "BB1") >= 0 && strcmp(parts[i].serial, "CC7") <= 0) {

printf("Serial: %s, Year: %d, Material: %s, Quantity: %d\n", parts[i].serial, parts[i].year, parts[i].material, parts[i].quantity);

}

}

return 0;

}

100. Write a program that receives an integer (less than or equal to nine digits in length) and prints out the number in words. For example, if the number input is 12342, then the output should be Twelve Thousand Three Hundred Forty Two

#include <stdio.h>

void print\_digit(int digit) {

const char \*words[] = {"Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};

printf("%s ", words[digit]);

}

void print\_number\_in\_words(int number) {

if (number >= 1000000000) {

print\_number\_in\_words(number / 1000000000);

printf("Billion ");

number %= 1000000000;

}

if (number >= 1000000) {

print\_number\_in\_words(number / 1000000);

printf("Million ");

number %= 1000000;

}

if (number >= 1000) {

print\_number\_in\_words(number / 1000);

printf("Thousand ");

number %= 1000;

}

if (number >= 100) {

print\_number\_in\_words(number / 100);

printf("Hundred ");

number %= 100;

}

if (number >= 20) {

const char \*words[] = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};

printf("%s ", words[number / 10]);

number %= 10;

}

if (number >= 10) {

const char \*words[] = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};

printf("%s ", words[number - 10]);

return;

}

if (number > 0) {

print\_digit(number);

}

}

int main() {

int number;

printf("Enter an integer number (less than or equal to nine digits): ");

scanf("%d", &number);

printf("Number in words: ");

if (number == 0) {

printf("Zero");

} else {

print\_number\_in\_words(number);

}

printf("\n");

return 0;

}