**KPIT TECHNOLOGIES**

**WEEKLY REPORT**

**WEEK 3- Report (DATE: 7/6/2024)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student name** | **Week** | **Branch** | **USN** |
| **Chinagani Tharun Sai Yadav** | **3** | **Circuit (ECE)** | **1NH20EC035** |

**Yashavant Kanetkar Book**

**Question 101-150:**

**Question 101: Attempt the following questions:**

**(a) If the string "Alice in wonder land" is fed to the following scanf( ) statement, what will be the contents of arrays str1, str2, str3 and str4?**

**scanf ( "%s%s%s%s", str1, str2, str3, str4 ) ;**

**(b) To uniquely identify a book a 10-digit ISBN (International Standard Book Number) is used. The rightmost digit in ISBN is a checksum digit. This digit is determined from the other 9 digits using the condition that d1 + 2d2 + 3d3 + ... + 10d10 must be a multiple of 11 (where di denotes the ith digit from the right). The checksum digit d1 can be any value from 0 to 10: the ISBN convention is to use the value X to denote 10. Write a program that receives a 10-digit integer, computes the checksum, and reports whether the ISBN number is correct or not.**

**(c) A Credit Card number is usually a 16-digit number. A valid Credit Card number would satisfy a rule explained below with the help of a dummy Credit Card number—4567 1234 5678 9129. Start with the rightmost - 1 digit and multiply every other digit by 2.**

**4 5 6 7 1 2 3 4 5 6 7 8 9 1 2 9**

**8 12 2 6 10 14 18 4**

**Then subtract 9 from numbers that are larger than 10. Thus, we get:**

**8 3 2 6 1 5 9 4**

**Add them all up to get 38.**

**Add all the other digits (5, 7, 2, 4, 6, 8, 1, 9) to get 42.**

**Sum of 38 and 42 is 80. Since 80 is divisible by 10, the Credit Card number is valid.**

**Write a program that receives a Credit Card number and checks using the above rule whether the Credit Card number is valid.**

**Sol.**

(a)

#include <stdio.h>

int main() {

char str1[10], str2[10], str3[10], str4[10];

char input[] = "Alice in wonder land";

sscanf(input, "%s %s %s %s", str1, str2, str3, str4);

printf("str1 = %s\n", str1);

printf("str2 = %s\n", str2);

printf("str3 = %s\n", str3);

printf("str4 = %s\n", str4);

return 0;

}"

(b)

#include <stdio.h>

#include <string.h>

int is\_valid\_isbn(char isbn[]) {

int length = strlen(isbn);

if (length != 10) {

return 0;

}

int total\_sum = 0;

for (int i = 0; i < 10; i++) {

int digit;

if (isbn[i] == 'X' && i == 9) { // 'X' is only valid in the last position

digit = 10;

} else if (isbn[i] >= '0' && isbn[i] <= '9') {

digit = isbn[i] - '0';

} else {

return 0; // invalid character

}

total\_sum += (10 - i) \* digit;

}

return total\_sum % 11 == 0;

}

int main() {

char isbn[11]; // 10 digits + null terminator

printf("Enter a 10-digit ISBN number: ");

scanf("%10s", isbn);

if (is\_valid\_isbn(isbn)) {

printf("The ISBN number %s is valid.\n", isbn);

} else {

printf("The ISBN number %s is not valid.\n", isbn);

}

return 0;

}

(c)

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int is\_valid\_credit\_card(char card\_number[]) {

int length = strlen(card\_number);

int sum = 0;

int is\_second = 0;

// Traverse the card number from right to left

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

if (!isdigit(card\_number[i])) {

continue; // Skip non-digit characters

}

int digit = card\_number[i] - '0';

if (is\_second) {

digit \*= 2;

if (digit > 9) {

digit -= 9;

}

}

sum += digit;

is\_second = !is\_second;

}

return sum % 10 == 0;

}

int main() {

char card\_number[17]; // 16 digits + null terminator

printf("Enter a 16-digit Credit Card number: ");

scanf("%16s", card\_number);

if (is\_valid\_credit\_card(card\_number)) {

printf("The Credit Card number %s is valid.\n", card\_number);

} else {

printf("The Credit Card number %s is not valid.\n", card\_number);

}

return 0;

}

**Question 102: Write a program which asks you to type your name. When you do so, it checks your name against a list of names to see if you are worthy of entry to the palace.**

**Sol.** #include <stdio.h>

#include <string.h>

int is\_worthy\_of\_entry(char \*name) {

const char \*worthy\_names[] = {

"Alice",

"Bob",

"Charlie",

"Diana",

"Edward"

};

int num\_names = sizeof(worthy\_names) / sizeof(worthy\_names[0]);

for (int i = 0; i < num\_names; i++) {

if (strcmp(name, worthy\_names[i]) == 0) {

return 1; // Name is found in the list

}

}

return 0; // Name is not found in the list

}

int main() {

char name[50];

printf("Please enter your name: ");

scanf("%49s", name); // Read up to 49 characters to leave space for the null terminator

if (is\_worthy\_of\_entry(name)) {

printf("Welcome to the palace, %s!\n", name);

} else {

printf("Sorry, %s, you are not worthy of entry to the palace.\n", name);

}

return 0;

}

**Question 103: 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.**

**Sol** #include <stdio.h>

#include <string.h>

int main() {

// Define an array of pointers to strings

const char \*string\_array[] = {

"Alice",

"Bob",

"Charlie",

"Diana",

"Edward"

};

int num\_strings = sizeof(string\_array) / sizeof(string\_array[0]);

char input[50];

// Prompt the user for a string

printf("Please enter a string: ");

scanf("%49s", input); // Read up to 49 characters to leave space for the null terminator

// Check if the input string is in the array

int found = 0;

for (int i = 0; i < num\_strings; i++) {

if (strcmp(input, string\_array[i]) == 0) {

found = 1;

break;

}

}

// Print the result

if (found) {

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

} else {

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

}

return 0;

}

Output

Please enter a string: Alice

The string 'Alice' is present in the array.

**Question 104: Write a program to alphabetically sort a set of names stored using an array of pointers to strings.**

**Sol**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Function to compare two strings, used by qsort

int compare\_strings(const void \*a, const void \*b) {

const char \*\*str1 = (const char \*\*)a;

const char \*\*str2 = (const char \*\*)b;

return strcmp(\*str1, \*str2);

}

int main() {

// Array of pointers to strings (names)

const char \*names[] = {

"Charlie",

"Alice",

"Edward",

"Bob",

"Diana"

};

int num\_names = sizeof(names) / sizeof(names[0]);

// Print names before sorting

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

for (int i = 0; i < num\_names; i++) {

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

}

// Sort the names using qsort

qsort(names, num\_names, sizeof(const char \*), compare\_strings);

// Print names after sorting

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

for (int i = 0; i < num\_names; i++) {

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

}

return 0;

}

Output:

Names before sorting:

Charlie

Alice

Edward

Bob

Diana

Names after sorting:

Alice

Bob

Charlie

Diana

Edward

**Question 105: Write a program to reverse the strings stored in an array of pointers to strings:**

**Sol.**

#include <stdio.h>

#include <string.h>

// Function to reverse a string

void reverse\_string(char \*str) {

int length = strlen(str);

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

char temp = str[i];

str[i] = str[length - i - 1];

str[length - i - 1] = temp;

}

}

int main() {

// Array of pointers to strings

const char \*strings[] = {

"Hello",

"World",

"OpenAI",

"ChatGPT"

};

int num\_strings = sizeof(strings) / sizeof(strings[0]);

// Print strings before reversing

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

for (int i = 0; i < num\_strings; i++) {

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

}

// Reverse the strings

for (int i = 0; i < num\_strings; i++) {

reverse\_string((char \*)strings[i]);

}

// Print strings after reversing

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

for (int i = 0; i < num\_strings; i++) {

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

}

return 0;

}

Output

Strings before reversing:

Hello

World

OpenAI

ChatGPT

Strings after reversing:

olleH

dlroW

IANepO

TPGtahC

**Question 106: Answer the following questions:**

**(a) How many bytes in memory would be occupied by the following array of pointers to strings? How many bytes would be required to store the same strings in a two-dimensional character array?**

**char \*mess[ ] = {**

**"Hammer and tongs", "Tooth and nail",**

**"Spit and polish", "You and C"**

**} ;**

**(b) Write a program to delete all vowels from a sentence. Assume that the sentence is not more than 80 characters long.**

**(c) Write a program that will read a line and delete from it all occurrences of the word ‘the’.**

**(d) Write a program that stores a set of names of individuals and abbreviates the first and middle name to their first letter.**

**(e) 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.**

**(f) 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.**

**Sol.** (a)

Memory occupied by char \*mess []: 16 bytes (on a 32-bit system) or 32 bytes (on a 64-bit system).

Memory required for two-dimensional character array: 57 bytes.

(b) #include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_LENGTH 80

// Function to delete vowels from a string

void delete\_vowels(char \*sentence) {

int length = strlen(sentence);

// Iterate through each character in the string

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

char ch = tolower(sentence[i]); // Convert character to lowercase

// Check if the character is a vowel (a, e, i, o, u)

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

// Shift characters to the left to remove the vowel

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

sentence[j] = sentence[j + 1];

}

sentence[length - 1] = '\0'; // Null terminate the string

length--; // Decrement the length of the string

i--; // Decrement i to account for the shifted characters

}

}

}

int main() {

char sentence[MAX\_LENGTH + 1]; // Add 1 for the null terminator

// Read the sentence from the user

printf("Enter a sentence (not more than 80 characters): ");

fgets(sentence, sizeof(sentence), stdin);

sentence[strcspn(sentence, "\n")] = '\0'; // Remove the newline character if present

// Delete vowels from the sentence

delete\_vowels(sentence);

// Print the modified sentence

printf("Sentence after removing vowels: %s\n", sentence);

return 0;

}

(c)

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_LENGTH 1000

void delete\_the(char \*line) {

char \*ptr = line;

int the\_length = strlen("the");

while ((ptr = strstr(ptr, "the")) != NULL) {

if ((ptr == line || !isalpha(\*(ptr - 1))) && !isalpha(\*(ptr + the\_length))) {

memmove(ptr, ptr + the\_length, strlen(ptr + the\_length) + 1);

} else {

ptr += the\_length;

}

}

}

int main() {

char line[MAX\_LENGTH];

printf("Enter a line: ");

fgets(line, sizeof(line), stdin);

line[strcspn(line, "\n")] = '\0';

delete\_the(line);

printf("Line after removing 'the': %s\n", line);

return 0;

}

(d)

#include <stdio.h>

#include <string.h>

#define MAX\_NAMES 5

#define MAX\_LENGTH 50

void abbreviate\_names(char names[][MAX\_LENGTH]) {

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

char \*token = strtok(names[i], " ");

if (token != NULL) {

printf("%c", \*token); // Print the first letter of the first name

token = strtok(NULL, " "); // Get the next token (middle name)

while (token != NULL) {

printf(".%c", \*token); // Print the first letter of the middle name

token = strtok(NULL, " "); // Get the next token (last name)

}

}

printf(" %s\n", strtok(NULL, "\n")); // Print the last name

}

}

int main() {

char names[MAX\_NAMES][MAX\_LENGTH];

printf("Enter %d names:\n", MAX\_NAMES);

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

fgets(names[i], sizeof(names[i]), stdin);

names[i][strcspn(names[i], "\n")] = '\0'; // Remove the newline character if present

}

printf("\nAbbreviated names:\n");

abbreviate\_names(names);

return 0;

}

(e)

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int count\_successive\_vowels(char \*line) {

int count = 0;

int length = strlen(line);

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

char current = tolower(line[i]);

char next = tolower(line[i + 1]);

if ((current == 'a' || current == 'e' || current == 'i' || current == 'o' || current == 'u') &&

(next == 'a' || next == 'e' || next == 'i' || next == 'o' || next == 'u')) {

count++;

printf("%c%c\n", current, next);

}

}

return count;

}

int main() {

char line[1000];

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

fgets(line, sizeof(line), stdin);

line[strcspn(line, "\n")] = '\0'; // Remove the newline character if present

int occurrences = count\_successive\_vowels(line);

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

return 0;

}

(f)

#include <stdio.h>

#include <string.h>

// Function to convert a two-digit number into words

void convert\_two\_digits(int num, char \*result) {

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

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

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

if (num >= 20) {

strcat(result, tens[num / 10]);

if (num % 10 != 0) {

strcat(result, " ");

strcat(result, ones[num % 10]);

}

} else if (num >= 10) {

strcat(result, teens[num - 10]);

} else {

strcat(result, ones[num]);

}

}

// Function to convert a three-digit number into words

void convert\_three\_digits(int num, char \*result) {

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

strcat(result, ones[num / 100]);

if (num / 100 != 0) {

strcat(result, " Hundred ");

}

num %= 100;

if (num != 0) {

convert\_two\_digits(num, result);

}

}

// Function to convert a number into words

void convert\_to\_words(long long num, char \*result) {

const char \*suffixes[] = {"", " Thousand ", " Million ", " Billion "};

if (num == 0) {

strcat(result, "Zero");

return;

}

int i = 0;

while (num > 0) {

int part = num % 1000;

if (part != 0) {

convert\_three\_digits(part, result);

strcat(result, suffixes[i]);

}

num /= 1000;

i++;

}

}

int main() {

long long num;

char result[1000] = "";

printf("Enter an integer (up to nine digits): ");

scanf("%lld", &num);

convert\_to\_words(num, result);

printf("Number in words: %s\n", result);

return 0;

}

**Question 107: Write a program where we wish to store in memory name (a string), price (a float) and number of pages (an int) of 3 books. To do this we can take following approaches:**

**(a) Construct 3 arrays for storing names, prices and number of pages.**

**(b) Use a structure variable.**

**Sol:**

A)

#include <stdio.h>

#include <string.h>

#define MAX\_BOOKS 3

#define MAX\_NAME\_LENGTH 50

int main() {

char names[MAX\_BOOKS][MAX\_NAME\_LENGTH];

float prices[MAX\_BOOKS];

int pages[MAX\_BOOKS];

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

printf("Enter name of book %d: ", i + 1);

fgets(names[i], sizeof(names[i]), stdin);

names[i][strcspn(names[i], "\n")] = '\0'; // Remove newline character

printf("Enter price of book %d: ", i + 1);

scanf("%f", &prices[i]);

printf("Enter number of pages of book %d: ", i + 1);

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

getchar(); // Consume newline character

}

printf("\nDetails of the books:\n");

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

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

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

printf("Price: %.2f\n", prices[i]);

printf("Number of pages: %d\n", pages[i]);

printf("\n");

}

return 0;

}

B)

#include <stdio.h>

#include <string.h>

#define MAX\_NAME\_LENGTH 50

struct Book {

char name[MAX\_NAME\_LENGTH];

float price;

int pages;

};

int main() {

struct Book books[3];

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

printf("Enter name of book %d: ", i + 1);

fgets(books[i].name, sizeof(books[i].name), stdin);

books[i].name[strcspn(books[i].name, "\n")] = '\0'; // Remove newline character

printf("Enter price of book %d: ", i + 1);

scanf("%f", &books[i].price);

printf("Enter number of pages of book %d: ", i + 1);

scanf("%d", &books[i].pages);

getchar(); // Consume newline character

}

printf("\nDetails of the books:\n");

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

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

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

printf("Price: %.2f\n", books[i].price);

printf("Number of pages: %d\n", books[i].pages);

printf("\n");

}

return 0;

}

**Question 108: 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.**

**Sol.**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node \*createNode(int data) {

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

if (newNode == NULL) {

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

exit(EXIT\_FAILURE);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void push(struct Node \*\*top, int data) {

struct Node \*newNode = createNode(data);

newNode->next = \*top;

\*top = newNode;

printf("%d pushed to stack.\n", data);

}

int pop(struct Node \*\*top) {

if (\*top == NULL) {

printf("Stack underflow!\n");

exit(EXIT\_FAILURE);

}

struct Node \*temp = \*top;

int data = temp->data;

\*top = (\*top)->next;

free(temp);

return data;

}

void display(struct Node \*top) {

printf("Stack: ");

while (top != NULL) {

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

top = top->next;

}

printf("\n");

}

int main() {

struct Node \*top = NULL;

push(&top, 10);

push(&top, 20);

push(&top, 30);

display(top);

printf("Popped element: %d\n", pop(&top));

printf("Popped element: %d\n", pop(&top));

display(top);

return 0;

}

Output

10 pushed to stack.

20 pushed to stack.

30 pushed to stack.

Stack: 30 20 10

Popped element: 30

Popped element: 20

Stack: 10

**Question 109: 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.**

**Sol.**

#include <stdio.h>

#include <stdlib.h>

int data;

struct Node \*next;

};

struct Queue {

struct Node \*front;

struct Node \*rear;

};

struct Node \*createNode(int data) {

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

if (newNode == NULL) {

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

exit(EXIT\_FAILURE);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

struct Queue \*createQueue() {

struct Queue \*queue = (struct Queue \*)malloc(sizeof(struct Queue));

if (queue == NULL) {

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

exit(EXIT\_FAILURE);

}

queue->front = NULL;

queue->rear = NULL;

return queue;

}

int isEmpty(struct Queue \*queue) {

return (queue->front == NULL);

}

void enqueue(struct Queue \*queue, int data) {

struct Node \*newNode = createNode(data);

if (isEmpty(queue)) {

queue->front = newNode;

queue->rear = newNode;

} else {

queue->rear->next = newNode;

queue->rear = newNode;

}

printf("%d enqueued to queue.\n", data);

}

int dequeue(struct Queue \*queue) {

if (isEmpty(queue)) {

printf("Queue underflow!\n");

exit(EXIT\_FAILURE);

}

struct Node \*temp = queue->front;

int data = temp->data;

queue->front = queue->front->next;

if (queue->front == NULL) {

queue->rear = NULL;

}

free(temp);

return data;

}

void display(struct Queue \*queue) {

if (isEmpty(queue)) {

printf("Queue is empty.\n");

return;

}

printf("Queue: ");

struct Node \*temp = queue->front;

while (temp != NULL) {

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

temp = temp->next;

}

printf("\n");

}

int main() {

struct Queue \*queue = createQueue();

enqueue(queue, 10);

enqueue(queue, 20);

enqueue(queue, 30);

display(queue);

printf("Dequeued element: %d\n", dequeue(queue));

printf("Dequeued element: %d\n", dequeue(queue));

display(queue);

return 0;

}

Output:

10 enqueued to queue.

20 enqueued to queue.

30 enqueued to queue.

Queue: 10 20 30

Dequeued element: 10

Dequeued element: 20

Queue: 30

**Question 110: Answer the following questions:**

**(a) Given the statement,**

**maruti.engine.bolts = 25 ; which of the following is True?**

**1. bolts is a structure variable**

**2. engine is a structure variable**

**3. maruti is a structure variable**

**4. Option 2. and 3.**

**(b) struct time**

**{**

**int hours ; int minutes ; int seconds ;**

**} t ;**

**struct time \*pt ;**

**pt = &t ;**

**With reference to the above declarations which of the following refers to seconds correctly:**

**1. pt.seconds**

**2. pt -> seconds**

**3. time.seconds**

**4. time->seconds**

**Sol.** (a) the correct statement would be:

4) Option 2. and 3.

(b) the correct choice is:

1. pt -> seconds

**Question 111: Attempt the following questions:**

**(a) 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.**

**(b) Create a structure that can contain data of customers in a bank. The data to be stored is Account number, Name and Balance in account.**

**Assume maximum of 200 customers in the bank.**

**(1) Define a function to print the Account number and name of each customer with balance below Rs. 1000.**

**(2) If a customer requests for withdrawal or deposit, it should receive as input Account number, amount and code (1 for deposit, 0 for withdrawal).**

**Define a function that prints a message, “The balance is insufficient for the specified withdrawal”, if on withdrawal the balance falls below Rs. 1000.**

**(c) 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.**

**(d) A record contains name of cricketer, his age, number of test matches that he has played and the average runs that he has scored. Create an array of structures to hold records of 20 such criketers and then write a program to read these records and arrange them in ascending order by average runs. Use the qsort( ) standard library function.**

**(e) 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.**

**(f) Create a structure called library to hold accession number, title of the book, author name, price of the book, and flag indicating whether book is issued or not. Write a menu-driven program that implements the working of a library. The menu options should be:**

**1. Add book information**

**2. Display book information**

**3. List all books of given author**

**4. List the title of book specified by accession number**

**5. List the count of books in the library**

**6. List the books in the order of accession number**

**7. Exit**

**Sol.**

(a) #include <stdio.h>

#include <string.h>

#define MAX\_STUDENTS 450

#define MAX\_NAME\_LENGTH 50

#define MAX\_DEPT\_LENGTH 50

#define MAX\_COURSE\_LENGTH 50

// Structure to represent a student

struct Student {

int rollNumber;

char name[MAX\_NAME\_LENGTH];

char department[MAX\_DEPT\_LENGTH];

char course[MAX\_COURSE\_LENGTH];

int yearOfJoining;

};

// Function to print names of students who joined in a particular year

void printStudentsByYear(struct Student students[], int numOfStudents, int year) {

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

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

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

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

}

}

}

// Function to print data of a student given their roll number

void printStudentByRollNumber(struct Student students[], int numOfStudents, int rollNumber) {

for (int i = 0; i < numOfStudents; 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);

}

int main() {

struct Student students[MAX\_STUDENTS] = {

{101, "John", "Computer Science", "B.Tech", 2019},

{102, "Alice", "Electrical Engineering", "B.E", 2020},

{103, "Bob", "Mechanical Engineering", "B.E", 2019},

// Add more students here...

};

int numOfStudents = 3; // Update this with the actual number of students

int yearToSearch = 2019;

printStudentsByYear(students, numOfStudents, yearToSearch);

int rollNumberToSearch = 102;

printStudentByRollNumber(students, numOfStudents, rollNumberToSearch);

return 0;

}

(b) #include <stdio.h>

#include <string.h>

#define MAX\_CUSTOMERS 200

#define MAX\_NAME\_LENGTH 50

// Structure to represent a customer in a bank

struct Customer {

int accountNumber;

char name[MAX\_NAME\_LENGTH];

float balance;

};

// Function to print account number and name of customers with balance below Rs. 1000

void printCustomersWithLowBalance(struct Customer customers[], int numOfCustomers) {

printf("Customers with balance below Rs. 1000:\n");

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

if (customers[i].balance < 1000) {

printf("Account Number: %d, Name: %s\n", customers[i].accountNumber, customers[i].name);

}

}

}

// Function to handle withdrawal or deposit for a customer

void handleTransaction(struct Customer customers[], int numOfCustomers, int accountNumber, float amount, int code) {

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

if (customers[i].accountNumber == accountNumber) {

if (code == 1) { // Deposit

customers[i].balance += amount;

printf("Deposit of Rs. %.2f successful.\n", amount);

} else if (code == 0) { // Withdrawal

if (customers[i].balance - amount < 1000) {

printf("The balance is insufficient for the specified withdrawal.\n");

} else {

customers[i].balance -= amount;

printf("Withdrawal of Rs. %.2f successful.\n", amount);

}

} else {

printf("Invalid code. Use 1 for deposit and 0 for withdrawal.\n");

}

return;

}

}

printf("Customer with Account Number %d not found.\n", accountNumber);

}

int main() {

struct Customer customers[MAX\_CUSTOMERS] = {

{101, "John", 1500},

{102, "Alice", 800},

{103, "Bob", 1200},

// Add more customers here...

};

int numOfCustomers = 3; // Update this with the actual number of customers

printCustomersWithLowBalance(customers, numOfCustomers);

int accountNumber = 102;

float amount = 200;

int code = 1; // 1 for deposit, 0 for withdrawal

handleTransaction(customers, numOfCustomers, accountNumber, amount, code);

return 0;

}

(c)

#include <stdio.h>

#include <string.h>

#define MAX\_PARTS 360 // Total number of parts (FF9 - AA0 + 1)

// Structure to represent information about a part

struct Part {

char serialNumber[4]; // Serial number (e.g., "AA0", "AA1", ..., "FF9")

int yearOfManufacture;

char material[20];

int quantityManufactured;

};

// Function to retrieve information on parts with serial numbers between startSN and endSN

void retrievePartsInfo(struct Part parts[], int numOfParts, char startSN[], char endSN[]) {

printf("Parts information between serial numbers %s and %s:\n", startSN, endSN);

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

if (strcmp(parts[i].serialNumber, startSN) >= 0 && strcmp(parts[i].serialNumber, endSN) <= 0) {

printf("Serial Number: %s\n", parts[i].serialNumber);

printf("Year of Manufacture: %d\n", parts[i].yearOfManufacture);

printf("Material: %s\n", parts[i].material);

printf("Quantity Manufactured: %d\n", parts[i].quantityManufactured);

printf("\n");

}

}

}

int main() {

// Create an array of Part structures and initialize with some sample data

struct Part parts[MAX\_PARTS];

int numOfParts = 0;

// Simulate data generation (in real scenario, this data may be read from a file or database)

for (char c1 = 'A'; c1 <= 'F'; c1++) {

for (char c2 = 'A'; c2 <= 'F'; c2++) {

for (int num = 0; num <= 9; num++) {

sprintf(parts[numOfParts].serialNumber, "%c%c%d", c1, c2, num);

parts[numOfParts].yearOfManufacture = 2022; // Sample year

strcpy(parts[numOfParts].material, "Steel"); // Sample material

parts[numOfParts].quantityManufactured = 100; // Sample quantity

numOfParts++;

}

}

}

// Retrieve information on parts with serial numbers between "BB1" and "CC6"

retrievePartsInfo(parts, numOfParts, "BB1", "CC6");

return 0;

}

(d)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_CRICKETERS 20

// Structure to represent information about a cricketer

struct Cricketer {

char name[50];

int age;

int testMatches;

float averageRuns;

};

// Function to compare average runs of two cricketers (used for qsort)

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

const struct Cricketer \*c1 = (const struct Cricketer \*)a;

const struct Cricketer \*c2 = (const struct Cricketer \*)b;

if (c1->averageRuns < c2->averageRuns) return -1;

if (c1->averageRuns > c2->averageRuns) return 1;

return 0;

}

int main() {

// Create an array of Cricketer structures to hold records of cricketers

struct Cricketer cricketers[MAX\_CRICKETERS];

// Read records of cricketers

printf("Enter details of %d cricketers:\n", MAX\_CRICKETERS);

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

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

printf("Name: ");

fgets(cricketers[i].name, sizeof(cricketers[i].name), stdin);

cricketers[i].name[strcspn(cricketers[i].name, "\n")] = '\0'; // Remove newline character

printf("Age: ");

scanf("%d", &cricketers[i].age);

printf("Number of test matches: ");

scanf("%d", &cricketers[i].testMatches);

printf("Average runs: ");

scanf("%f", &cricketers[i].averageRuns);

getchar(); // Clear input buffer

}

// Sort cricketers by average runs in ascending order using qsort

qsort(cricketers, MAX\_CRICKETERS, sizeof(struct Cricketer), compareCricketers);

// Display sorted records

printf("\nRecords of cricketers sorted by average runs in ascending order:\n");

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

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

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

printf("Age: %d\n", cricketers[i].age);

printf("Number of test matches: %d\n", cricketers[i].testMatches);

printf("Average runs: %.2f\n", cricketers[i].averageRuns);

printf("\n");

}

return 0;

}

(e)

#include <stdio.h>

#include <stdbool.h>

#define MAX\_EMPLOYEES 50

// Structure to represent information about an employee

struct Employee {

int employeeCode;

char name[50];

int joinYear;

};

// Function to calculate tenure in years

int calculateTenure(int currentYear, int joinYear) {

return currentYear - joinYear;

}

int main() {

// Create an array of Employee structures to hold information about employees

struct Employee employees[MAX\_EMPLOYEES];

// Read data of employees

int numEmployees;

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

scanf("%d", &numEmployees);

printf("Enter data of employees:\n");

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

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

printf("Employee Code: ");

scanf("%d", &employees[i].employeeCode);

printf("Name: ");

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

printf("Year of Joining: ");

scanf("%d", &employees[i].joinYear);

}

// Read current date

int currentYear;

printf("\nEnter current year: ");

scanf("%d", &currentYear);

// Display names of employees with tenure greater than or equal to 3 years

printf("\nEmployees with tenure greater than or equal to 3 years:\n");

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

int tenure = calculateTenure(currentYear, employees[i].joinYear);

if (tenure >= 3) {

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

}

}

return 0;

}

(f)

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#define MAX\_BOOKS 100

// Structure to represent information about a book

struct Book {

int accessionNumber;

char title[100];

char author[50];

float price;

bool isIssued;

};

// Structure to represent the library

struct Library {

struct Book books[MAX\_BOOKS];

int numBooks;

};

// Function prototypes

void addBook(struct Library \*library);

void displayBook(struct Library library);

void listBooksByAuthor(struct Library library, char author[]);

void listBookByAccessionNumber(struct Library library, int accessionNumber);

void listBookCount(struct Library library);

void listBooksByAccessionNumber(struct Library library);

void exitProgram();

int main() {

struct Library library = { .numBooks = 0 }; // Initialize the library

int choice;

do {

// Display menu

printf("\nLibrary Management System Menu:\n");

printf("1. Add book information\n");

printf("2. Display book information\n");

printf("3. List all books of given author\n");

printf("4. List the title of book specified by accession number\n");

printf("5. List the count of books in the library\n");

printf("6. List the books in the order of accession number\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

// Perform action based on user's choice

switch (choice) {

case 1:

addBook(&library);

break;

case 2:

displayBook(library);

break;

case 3: {

char author[50];

printf("Enter author name: ");

scanf("%s", author);

listBooksByAuthor(library, author);

break;

}

case 4: {

int accessionNumber;

printf("Enter accession number: ");

scanf("%d", &accessionNumber);

listBookByAccessionNumber(library, accessionNumber);

break;

}

case 5:

listBookCount(library);

break;

case 6:

listBooksByAccessionNumber(library);

break;

case 7:

exitProgram();

break;

default:

printf("Invalid choice! Please enter a number between 1 and 7.\n");

}

} while (choice != 7);

return 0;

}

// Function to add book information to the library

void addBook(struct Library \*library) {

if (library->numBooks < MAX\_BOOKS) {

printf("Enter book information:\n");

printf("Accession Number: ");

scanf("%d", &library->books[library->numBooks].accessionNumber);

printf("Title: ");

scanf("%s", library->books[library->numBooks].title);

printf("Author: ");

scanf("%s", library->books[library->numBooks].author);

printf("Price: ");

scanf("%f", &library->books[library->numBooks].price);

printf("Is Issued? (0 for No, 1 for Yes): ");

scanf("%d", &library->books[library->numBooks].isIssued);

library->numBooks++;

printf("Book added successfully.\n");

} else {

printf("Library is full. Cannot add more books.\n");

}

}

// Function to display book information

void displayBook(struct Library library) {

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

for (int i = 0; i < library.numBooks; i++) {

printf("Accession Number: %d, Title: %s, Author: %s, Price: %.2f, Issued: %s\n",

library.books[i].accessionNumber,

library.books[i].title,

library.books[i].author,

library.books[i].price,

library.books[i].isIssued ? "Yes" : "No");

}

}

// Function to list all books of a given author

void listBooksByAuthor(struct Library library, char author[]) {

printf("\nBooks by author '%s':\n", author);

for (int i = 0; i < library.numBooks; i++) {

if (strcmp(library.books[i].author, author) == 0) {

printf("Title: %s\n", library.books[i].title);

}

}

}

// Function to list the title of book specified by accession number

void listBookByAccessionNumber(struct Library library, int accessionNumber) {

bool found = false;

for (int i = 0; i < library.numBooks; i++) {

if (library.books[i].accessionNumber == accessionNumber) {

printf("Title of book with accession number %d: %s\n", accessionNumber, library.books[i].title);

found = true;

break;

}

}

if (!found) {

printf("Book with accession number %d not found.\n", accessionNumber);

}

}

// Function to list the count of books in the library

void listBookCount(struct Library library) {

printf("\nNumber of books in the library: %d\n", library.numBooks);

}

// Function to list the books in the order of accession number

void listBooksByAccessionNumber(struct Library library) {

printf("\nBooks in the order of accession number:\n");

for (int i = 0; i < library.numBooks; i++) {

printf("Accession Number: %d, Title: %s\n",

library.books[i].accessionNumber,

library.books[i].title);

}

}

// Function to exit the program

void exitProgram() {

printf("Exiting the program. Goodbye!\n");

exit(0);

}

**Question 112: What will be the output of the following programs?**

**(a) # include <stdio.h>**

**# include <ctype.h>**

**int main( )**

**{**

**char ch ;**

**ch = getchar( ) ;**

**if ( islower ( ch ) )**

**putchar ( toupper ( ch ) ) ;**

**else**

**putchar ( tolower ( ch ) ) ;**

**return 0 ;**

**}**

**(b) # include <stdio.h>**

**int main( )**

**{**

**int i = 2 ;**

**float f = 2.5367 ;**

**char str[ ] = "Life is like that" ;**

**printf ( "%4d\t%3.3f\t%4s\n", i, f, str ) ;**

**return 0 ;**

**}**

**(c) # include <stdio.h>**

**int main( )**

**{**

**printf ( "More often than \b\b not \rthe person who \**

**wins is the one who thinks he can!\n" ) ;**

**return 0 ;**

**}**

**(d) # include <conio.h>**

**char p[ ] = "The sixth sick sheikh's sixth ship is sick" ;**

**int main( )**

**{**

**int i = 0 ;**

**while ( p[ i ] != '\0' )**

**{**

**putchar ( p[ i ] ) ;**

**i++ ;**

**}**

**return 0 ;**

**}**

**Sol.** (a) If the input character is 'a', the output will be 'A'.

If the input character is 'Z', the output will be 'z'.

If the input character is '7', it won't be affected by the case conversion and will be printed as is.

(b) 2 2.537 Life is like that

(c) the person who wins is the one who thinks he can!

(d) The sixth sick sheikh's sixth ship is sick.

**Question 113: Point out the errors, if any, in the following programs:**

**(a) # include <stdio.h>**

**int main( )**

**{**

**int i ;**

**char a[ ] = "Hello" ;**

**while ( a != '\0' )**

**{**

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

**a++ ;**

**}**

**return 0 ;**

**}**

**(b) # include <stdio.h>**

**int main( )**

**{**

**double dval ;**

**scanf ( "%f", &dval ) ;**

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

**return 0 ;**

**}**

**(c) # include <stdio.h>**

**int main( )**

**{**

**int ival ;**

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

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

**return 0 ;**

**}**

**(d) # 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 ;**

**}**

**(e) # 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 ;**

**}**

**(f) # include <stdio.h>**

**int main( )**

**{**

**char buffer[ 50 ] ;**

**int no = 97;**

**double val = 2.34174 ;**

**char name[ 10 ] = "Shweta" ;**

**sprintf ( buffer, "%d %lf %s", no, val, name ) ;**

**printf ( "%s\n", buffer ) ;**

**sscanf ( buffer, "%4d %2.2lf %s", &no, &val, name ) ;**

**printf ( "%s\n", buffer ) ;**

**printf ( "%d %lf %s\n", no, val, name ) ;**

**return 0 ;**

**}**

**Sol.** (a) The comparison a != '\0' is incorrect because a is an array and cannot be directly compared to a character. The intention is to check the end of the string.

a is an array, and arrays in C are not modifiable l-values; you cannot increment the array pointer directly. Instead, use a pointer to iterate through the array.

(b) The format specifier %f in scanf is for float, not double. For double, use %lf.

(c) The variable n is not defined. It should be ival.

The \n in scanf is unnecessary and can cause the function to wait for extra input.

(d) There are no errors in this code. It correctly uses %\*c to skip the delimiter characters / or -

(e) text should be a character array, not a single character. sprintf requires the first argument to be a pointer to a buffer large enough to hold the formatted string.

(f) The sscanf format specifiers %4d %2.2lf %s are incorrect because %4d expects exactly 4 digits and %2.2lf expects exactly 2 digits before and after the decimal point. These might not match the sprintf output correctly.

The format specifiers for sscanf should match the format specifiers used in sprintf.

**Question 114: Answer the following questions:**

**(a) To receive the string "We have got the guts, you get the glory!!" in**

**an array char str[ 100 ] which of the following functions would you**

**use?**

**1. scanf ( "%s", str ) ;**

**2. gets ( str ) ;**

**3. getchar ( str ) ;**

**4. fgetchar ( str ) ;**

**(b) If an integer is to be entered through the keyboard, which function**

**would you use?**

**1. scanf( )**

**2. gets( )**

**3. getche( )**

**4. getchar( )**

**(c) Which of the following can a format string of a printf( ) function**

**contain:**

**1. Characters, format specifications and escape sequences**

**2. Character, integers and floats**

**3. Strings, integers and escape sequences**

**4. Inverted commas, percentage sign and backslash character**

**(d) The purpose of the field-width specifier in a printf( ) function is to:**

**1. Control the margins of the program listing**

**2. Specify the maximum value of a number**

**3. Control the size of font used to print numbers**

**4. Specify how many columns should be used to print the number**

**(e) If we are to display the following output properly aligned which**

**format specifiers would you use?**

**Discovery of India Jawaharlal Nehru 425.50**

**My Experiments with Truth Mahatma Gandhi 375.50**

**Sunny Days Sunil Gavaskar 95.50**

**One More Over Erapalli Prasanna 85.00**

**Sol.** (a) gets(str);

(b) scanf()

(c) Characters, format specifications, and escape sequences

(d) Specify how many columns should be used to print the number

(e) You would use a combination of format specifiers with specified field widths to align the output.

**Question 115: Let us write a program to read a file and display its contents on the screen.**

**Sol.**

#include <stdio.h>

int main() {

FILE \*file;

char filename[100];

char line[1000];

printf("Enter the filename: ");

scanf("%s", filename);

file = fopen(filename, "r");

if (file == NULL) {

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

return 1;

}

printf("Contents of the file %s:\n", filename);

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

printf("%s", line);

}

fclose(file);

return 0;

}

**Question 116: let us write a program that will read a file and count how many characters, spaces, tabs and newlines are present in it.**

**Sol.**

#include <stdio.h>

int main() {

FILE \*file;

char filename[100];

char ch;

int charCount = 0, spaceCount = 0, tabCount = 0, newlineCount = 0;

printf("Enter the filename: ");

scanf("%s", filename);

file = fopen(filename, "r");

if (file == NULL) {

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

return 1;

}

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

charCount++;

if (ch == ' ')

spaceCount++;

else if (ch == '\t')

tabCount++;

else if (ch == '\n')

newlineCount++;

}

fclose(file);

printf("Total characters: %d\n", charCount);

printf("Total spaces: %d\n", spaceCount);

printf("Total tabs: %d\n", tabCount);

printf("Total newlines: %d\n", newlineCount);

return 0;

}

OUTPUT:

Total characters: 61

Total spaces: 9

Total tabs: 2

Total newlines: 2

**Question 117: Write a program to demonstrate the practical use of these character I/O functions**

**Sol.**

#include <stdio.h>

int main() {

char ch;

int charCount = 0;

printf("Enter some text (press Enter to finish):\n");

while ((ch = getchar()) != '\n') {

putchar(ch); // Echo the character to the screen

charCount++; // Increment character count

}

printf("\nTotal characters entered: %d\n", charCount);

return 0;

}

**Question 118: Write a program that writes strings to a file using fputs( ) and then reads them back using fgets( ).**

**Sol.**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file;

char filename[] = "strings.txt";

char str[100];

file = fopen(filename, "w");

if (file == NULL) {

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

return 1;

}

fputs("Hello, world!\n", file);

fputs("This is a test.\n", file);

fputs("Goodbye!\n", file);

fclose(file);

file = fopen(filename, "r");

if (file == NULL) {

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

return 1;

}

printf("Strings read from the file:\n");

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

printf("%s", str);

}

fclose(file);

return 0;

}

**Question 119: Write a program to record the I/O functions in files.**

**Sol.**

#include <stdio.h>

int main() {

FILE \*inputFile, \*outputFile;

char filename[100];

char inputStr[100], outputStr[100];

printf("Enter the input filename: ");

scanf("%s", filename);

inputFile = fopen(filename, "r");

if (inputFile == NULL) {

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

return 1;

}

printf("Enter the output filename: ");

scanf("%s", filename);

outputFile = fopen(filename, "w");

if (outputFile == NULL) {

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

fclose(inputFile);

return 1;

}

while (fgets(inputStr, sizeof(inputStr), inputFile) != NULL) {

fprintf(outputFile, "%s", inputStr);

}

fclose(inputFile);

fclose(outputFile);

printf("Input/output recorded successfully.\n");

return 0;

}

**Question 120: Write a program that performs all the file operations.**

**Sol.**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file;

char filename[100], inputStr[100], outputStr[100];

// Create a new file

printf("Enter the filename to create: ");

scanf("%s", filename);

file = fopen(filename, "w");

if (file == NULL) {

printf("Error creating the file.\n");

return 1;

}

printf("File created successfully: %s\n", filename);

fclose(file);

// Open the file for writing

printf("Enter some text to write to the file: ");

scanf("%s", inputStr);

file = fopen(filename, "w");

if (file == NULL) {

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

return 1;

}

fprintf(file, "%s\n", inputStr);

printf("Text written to the file.\n");

fclose(file);

// Open the file for reading and display its contents

file = fopen(filename, "r");

if (file == NULL) {

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

return 1;

}

printf("Contents of the file:\n");

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

printf("%s", outputStr);

}

fclose(file);

// Append text to the file

printf("Enter some text to append to the file: ");

scanf("%s", inputStr);

file = fopen(filename, "a");

if (file == NULL) {

printf("Error opening the file for appending.\n");

return 1;

}

fprintf(file, "%s\n", inputStr);

printf("Text appended to the file.\n");

fclose(file);

// Open the file for reading and display its contents again

file = fopen(filename, "r");

if (file == NULL) {

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

return 1;

}

printf("Contents of the file after appending:\n");

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

printf("%s", outputStr);

}

fclose(file);

// Delete the file

if (remove(filename) == 0) {

printf("File deleted successfully.\n");

} else {

printf("Error deleting the file.\n");

return 1;

}

return 0;

}

**Question 121: Write a program to read a file and display its contents along with line numbers before each line.**

**Sol.**

#include <stdio.h>

int main() {

FILE \*file;

char filename[100], line[1000];

int lineNum = 1;

// Prompt the user to enter the filename

printf("Enter the filename: ");

scanf("%s", filename);

// Open the file for reading

file = fopen(filename, "r");

// Check if the file opened successfully

if (file == NULL) {

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

return 1;

}

// Read each line from the file and display with line numbers

printf("Contents of the file %s:\n", filename);

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

printf("%d: %s", lineNum, line);

lineNum++;

}

// Close the file

fclose(file);

return 0;

}

Output:

Enter the filename: example.txt

Contents of the file example.txt:

1: This is line 1.

2: This is line 2.

3: This is line 3.

**Question 122: Write a program to append the contents of one file at the end of another.**

**Sol.**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*sourceFile, \*destinationFile;

char sourceFilename[100], destinationFilename[100];

char ch;

printf("Enter the source filename: ");

scanf("%s", sourceFilename);

printf("Enter the destination filename: ");

scanf("%s", destinationFilename);

sourceFile = fopen(sourceFilename, "r");

if (sourceFile == NULL) {

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

return 1;

}

destinationFile = fopen(destinationFilename, "a");

if (destinationFile == NULL) {

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

fclose(sourceFile);

return 1;

}

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

fputc(ch, destinationFile);

}

fclose(sourceFile);

fclose(destinationFile);

printf("Contents appended successfully.\n");

return 0;

}

Output:

Enter the source filename: source.txt

Enter the destination filename: destination.txt

Contents appended successfully.

**Question 123: Answer the following questions:**

**(a) In which file FILE structure is defined?**

**(b) If a file contains the line “I am a boy\r\n” then on reading this line into the array str[ ] using fgets( ) what would str[ ] contain?**

**(c) State True or False:**

**1. The disadvantage of high-level file I/O functions is that the programmer has to manage the file buffers.**

**2. If a file is opened for reading, it is necessary that the file must exist.**

**3. If a file opened for writing already exists, its contents would be overwritten.**

**4. For opening a file in append mode it is necessary that the file should exist.**

**(d) On opening a file for reading which of the following activities are performed:**

**1. The disk is searched for existence of the file.**

**2. The file contents are brought into memory.**

**3. A pointer is set up which points to the first character in the file.**

**4. All the above.**

**(e) Is it necessary that a file created in text mode must always be opened in text mode for subsequent operations?**

**Sol.** (a)The FILE structure is defined in the header file <stdio.h>.

(b) "I am a boy\r\n"

(c) False.

True

True

False

(d) The disk is searched for existence of the file.

A pointer is set up which points to the first character in the file.

(e) No, it is not necessary.

**Question 124: Attempt the following questions:**

**(a) 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.**

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

**(c) 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.**

**(d) Write a program to encrypt/decrypt a file using:**

**(1) Offset cipher: In this cipher each character from the source file is offset with a fixed value and then written to the target file.**

**For example, if character read from the source file is ‘A’, then write a character represented by ‘A’ + 128 to the target file.**

**(2) Substitution cipher: In this cipher for each character read from the source file a corresponding predetermined character is written to the target file.**

**For example, if character ‘A’ is read from the source file, then a ‘!’ would be written to the target file. Similarly, every ‘B’ would be substituted by ‘5’ and so on.**

**(e) In the file ‘CUSTOMER.DAT’ there are 10 records with the following structure:**

**struct customer**

**{**

**int accno ; char name[ 30 ] ; float balance ;**

**} ;**

**In another file ‘TRANSACTIONS.DAT’ there are several records with the following structure:**

**struct trans**

**{**

**int accno ; char trans\_type ; float amount ;**

**} ;**

**The element trans\_type contains D/W indicating deposit or withdrawal of amount. Write a program to update ‘CUSTOMER.DAT’ file, i.e., if the trans\_type is ‘D’ then update the balance of ‘CUSTOMER.DAT’ by adding amount to balance for the corresponding accno. Similarly, if trans\_type is ‘W’ then subtract the amount from balance. However, while subtracting the amount ensure that the amount should not get overdrawn, i.e., at least 100 Rs. should remain in the account.**

**(f) There are 10 records present in a file with the following structure:**

**struct date { int d, m, y ; } ;**

**struct employee**

**{**

**int empcode[ 6 ] ; char empname[ 20 ] ;**

**struct date join\_date ; float salary ;**

**} ;**

**Write a program to read these records, arrange them in ascending order by join\_date and write them to a target file.**

**(g) A hospital keeps a file of blood donors in which each record has the format:**

**Name: 20 columns Address: 40 columns**

**Age: 2 columns Blood Type: 1 column (Type 1, 2, 3 or 4)**

**Write a program to read the file and print a list of all blood donors whose age is below 25 and whose blood type is 2.**

**(h) Given a list of names of students in a class, write a program to store the names in a file on disk. Make a provision to display the nth name in the list, where n is read from the keyboard.**

**(i) Assume that a Master file contains two fields—roll number and name of the student. At the end of the year, a set of students join the class and another set leaves. A Transaction file contains the roll numbers and an appropriate code to add or delete a student.**

**Write a program to create another file that contains the updated list of names and roll numbers. Assume that the Master file and the Transaction file are arranged in ascending order by roll numbers.**

**The updated file should also be in ascending order by roll numbers.**

**(j) 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.**

**Sol.**

a) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_STUDENTS 100

struct Student {

char name[50];

int age;

};

int compareByName(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 numStudents = 0;

printf("Enter the filename: ");

scanf("%s", filename);

file = fopen(filename, "r");

if (file == NULL) {

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

return 1;

}

while (fscanf(file, "%s %d", students[numStudents].name, &students[numStudents].age) == 2) {

numStudents++;

}

fclose(file);

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

printf("Student records sorted by name:\n");

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

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

}

return 0;

}

(b)

#include <stdio.h>

#include <ctype.h>

int main() {

FILE \*sourceFile, \*destinationFile;

char sourceFilename[100], destinationFilename[100];

char ch;

printf("Enter the source filename: ");

scanf("%s", sourceFilename);

printf("Enter the destination filename: ");

scanf("%s", destinationFilename);

sourceFile = fopen(sourceFilename, "r");

if (sourceFile == NULL) {

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

return 1;

}

destinationFile = fopen(destinationFilename, "w");

if (destinationFile == NULL) {

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

fclose(sourceFile);

return 1;

}

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

if (islower(ch)) {

ch = toupper(ch);

}

fputc(ch, destinationFile);

}

fclose(sourceFile);

fclose(destinationFile);

printf("Contents copied successfully.\n");

return 0;

}

(c)

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file1, \*file2, \*outputFile;

char filename1[100], filename2[100], outputFilename[100];

char line1[1000], line2[1000];

printf("Enter the filename of the first input file: ");

scanf("%s", filename1);

printf("Enter the filename of the second input file: ");

scanf("%s", filename2);

file1 = fopen(filename1, "r");

file2 = fopen(filename2, "r");

if (file1 == NULL || file2 == NULL) {

printf("Error opening one of the input files.\n");

return 1;

}

printf("Enter the filename of the output file: ");

scanf("%s", outputFilename);

outputFile = fopen(outputFilename, "w");

if (outputFile == NULL) {

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

fclose(file1);

fclose(file2);

return 1;

}

while (fgets(line1, sizeof(line1), file1) || fgets(line2, sizeof(line2), file2)) {

if (line1[0] != '\0') {

fputs(line1, outputFile);

}

if (line2[0] != '\0') {

fputs(line2, outputFile);

}

}

fclose(file1);

fclose(file2);

fclose(outputFile);

printf("Lines merged successfully and written to %s.\n", outputFilename);

return 0;

}

(d) #include <stdio.h>

void offsetCipher(FILE \*sourceFile, FILE \*targetFile, int offset) {

char ch;

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

ch = ch + offset;

fputc(ch, targetFile);

}

}

void substitutionCipher(FILE \*sourceFile, FILE \*targetFile, char \*substitutionTable) {

char ch;

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

char \*ptr = substitutionTable;

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

if (\*ptr == ch) {

fputc(\*(ptr + 1), targetFile);

break;

}

ptr += 2; // Move to the next pair of characters in the table

}

}

}

int main() {

FILE \*sourceFile, \*targetFile;

char sourceFilename[100], targetFilename[100];

int choice, offset;

char substitutionTable[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*()";

printf("Enter the filename of the source file: ");

scanf("%s", sourceFilename);

printf("Enter the filename of the target file: ");

scanf("%s", targetFilename);

sourceFile = fopen(sourceFilename, "r");

if (sourceFile == NULL) {

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

return 1;

}

targetFile = fopen(targetFilename, "w");

if (targetFile == NULL) {

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

fclose(sourceFile);

return 1;

}

printf("Choose a cipher:\n");

printf("1. Offset cipher\n");

printf("2. Substitution cipher\n");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the offset value: ");

scanf("%d", &offset);

offsetCipher(sourceFile, targetFile, offset);

printf("File encrypted/decrypted using offset cipher.\n");

break;

case 2:

substitutionCipher(sourceFile, targetFile, substitutionTable);

printf("File encrypted/decrypted using substitution cipher.\n");

break;

default:

printf("Invalid choice.\n");

}

fclose(sourceFile);

fclose(targetFile);

return 0;

}

e)#include <stdio.h>

struct customer {

int accno;

char name[30];

float balance;

};

struct trans {

int accno;

char trans\_type;

float amount;

};

int main() {

FILE \*customerFile, \*transactionFile;

struct customer cust;

struct trans tr;

customerFile = fopen("CUSTOMER.DAT", "r+");

if (customerFile == NULL) {

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

return 1;

}

transactionFile = fopen("TRANSACTIONS.DAT", "r");

if (transactionFile == NULL) {

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

fclose(customerFile);

return 1;

}

while (fread(&tr, sizeof(struct trans), 1, transactionFile)) {

fseek(customerFile, (tr.accno - 1) \* sizeof(struct customer), SEEK\_SET);

fread(&cust, sizeof(struct customer), 1, customerFile);

if (tr.trans\_type == 'D') {

cust.balance += tr.amount;

} else if (tr.trans\_type == 'W') {

if (cust.balance - tr.amount >= 100) {

cust.balance -= tr.amount;

} else {

printf("Account %d does not have sufficient balance for withdrawal.\n", cust.accno);

continue; // Skip to the next transaction

}

}

fseek(customerFile, (tr.accno - 1) \* sizeof(struct customer), SEEK\_SET);

fwrite(&cust, sizeof(struct customer), 1, customerFile);

}

fclose(customerFile);

fclose(transactionFile);

printf("Customer records updated successfully.\n");

return 0;

}

(f) #include <stdio.h>

struct date {

int d, m, y;

};

struct employee {

int empcode[6];

char empname[20];

struct date join\_date;

float salary;

};

int compareDates(struct date d1, struct date d2) {

if (d1.y != d2.y) {

return d1.y - d2.y;

} else if (d1.m != d2.m) {

return d1.m - d2.m;

} else {

return d1.d - d2.d;

}

}

void swap(struct employee \*a, struct employee \*b) {

struct employee temp = \*a;

\*a = \*b;

\*b = temp;

}

void sortEmployees(struct employee arr[], int n) {

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

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

if (compareDates(arr[j].join\_date, arr[j + 1].join\_date) > 0) {

swap(&arr[j], &arr[j + 1]);

}

}

}

}

int main() {

FILE \*inputFile, \*outputFile;

struct employee employees[10];

inputFile = fopen("input\_file.dat", "rb");

if (inputFile == NULL) {

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

return 1;

}

fread(employees, sizeof(struct employee), 10, inputFile);

fclose(inputFile);

sortEmployees(employees, 10);

outputFile = fopen("output\_file.dat", "wb");

if (outputFile == NULL) {

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

return 1;

}

fwrite(employees, sizeof(struct employee), 10, outputFile);

fclose(outputFile);

printf("Employee records sorted and written to output file.\n");

return 0;

}

(g)

#include <stdio.h>

#include <string.h>

struct donor {

char name[21];

char address[41];

int age;

int blood\_type;

};

int main() {

FILE \*file;

struct donor donor\_record;

file = fopen("blood\_donors.txt", "r");

if (file == NULL) {

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

return 1;

}

printf("Blood donors below 25 years old with blood type 2:\n");

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

while (fscanf(file, "Name: %20s Address: %40s Age: %d Blood Type: %d\n",

donor\_record.name, donor\_record.address, &donor\_record.age, &donor\_record.blood\_type) != EOF) {

if (donor\_record.age < 25 && donor\_record.blood\_type == 2)

{

printf("Name: %-20s Address: %-40s Age: %2d Blood Type: %d\n",donor\_record.name, donor\_record.address, donor\_record.age, donor\_record.blood\_type);

}

}

fclose(file);

return 0;

}

(h)

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file;

char name[50];

int n, position;

file = fopen("student\_names.txt", "w");

if (file == NULL) {

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

return 1;

}

printf("Enter the names of students (enter 'quit' to stop):\n");

while (1) {

printf("> ");

scanf("%s", name);

if (strcmp(name, "quit") == 0) {

break;

}

fprintf(file, "%s\n", name);

}

fclose(file);

file = fopen("student\_names.txt", "r");

if (file == NULL) {

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

return 1;

}

printf("Enter the position of the name to display: ");

scanf("%d", &n);

position = (n - 1) \* 51; // Each name plus newline character takes 51 bytes

fseek(file, position, SEEK\_SET);

printf("The name at position %d is: ", n);

fgets(name, sizeof(name), file);

printf("%s", name);

fclose(file);

return 0;

}

(i) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct student {

int rollno;

char name[30];

};

struct transaction {

int rollno;

char code;

};

int main() {

struct student master[100], updated[100];

struct transaction trans;

int master\_count = 0, updated\_count = 0;

FILE \*masterFile = fopen("MASTER.DAT", "r");

if (masterFile == NULL) {

perror("Error opening master file");

return EXIT\_FAILURE;

}

while (fscanf(masterFile, "%d %s", &master[master\_count].rollno, master[master\_count].name) != EOF) {

master\_count++;

}

fclose(masterFile);

FILE \*transFile = fopen("TRANSACTION.DAT", "r");

if (transFile == NULL) {

perror("Error opening transaction file");

return EXIT\_FAILURE;

}

while (fscanf(transFile, "%d %c", &trans.rollno, &trans.code) != EOF) {

if (trans.code == 'A') {

master[master\_count].rollno = trans.rollno;

strcpy(master[master\_count].name, "New Student");

master\_count++;

} else if (trans.code == 'D') {

for (int i = 0; i < master\_count; i++) {

if (master[i].rollno == trans.rollno) {

for (int j = i; j < master\_count - 1; j++) {

master[j] = master[j + 1];

}

master\_count--;

break;

}

}

}

}

fclose(transFile);

masterFile = fopen("UPDATED\_MASTER.DAT", "w");

if (masterFile == NULL) {

perror("Error opening updated master file");

return EXIT\_FAILURE;

}

for (int i = 0; i < master\_count; i++) {

fprintf(masterFile, "%d %s\n", master[i].rollno, master[i].name);

}

fclose(masterFile);

return EXIT\_SUCCESS;

}

(j) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

int is\_word\_to\_delete(const char \*word) {

return strcmp(word, "a") == 0 || strcmp(word, "the") == 0 || strcmp(word, "an") == 0;

}

void process\_file(const char \*source, const char \*target) {

FILE \*srcFile, \*tgtFile;

char word[128];

srcFile = fopen(source, "r");

if (srcFile == NULL) {

perror("Error opening source file");

exit(EXIT\_FAILURE);

}

tgtFile = fopen(target, "w");

if (tgtFile == NULL) {

perror("Error opening target file");

fclose(srcFile);

exit(EXIT\_FAILURE);

}

while (fscanf(srcFile, "%127s", word) == 1) {

if (!is\_word\_to\_delete(word)) {

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

} else {

fprintf(tgtFile, " ");

}

}

fclose(srcFile);

fclose(tgtFile);

}

int main() {

char source[128], target[128];

printf("Enter source file name: ");

scanf("%s", source);

printf("Enter target file name: ");

scanf("%s", target);

process\_file(source, target);

return EXIT\_SUCCESS;

}

**Question 125: Write a program where Instead of the program prompting us to enter these filenames, we should be able to supply them at command prompt, in the form: filecopy PR1.C PR2.C**

**Sol.**

#include <stdio.h>

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

FILE \*sourceFile, \*destinationFile;

char ch;

if (argc != 3) {

printf("Usage: %s <source\_file> <destination\_file>\n", argv[0]);

return 1;

}

sourceFile = fopen(argv[1], "r");

if (sourceFile == NULL) {

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

return 1;

}

destinationFile = fopen(argv[2], "w");

if (destinationFile == NULL) {

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

fclose(sourceFile);

return 1;

}

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

fputc(ch, destinationFile);

}

fclose(sourceFile);

fclose(destinationFile);

printf("File copied successfully.\n");

return 0;

}

**Question 126: Write a program where there must be a provision to test whether our attempt to read/write was successful or not.**

**Sol.**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*sourceFile, \*destinationFile;

char ch;

sourceFile = fopen("source.txt", "r");

if (sourceFile == NULL) {

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

return 1;

}

destinationFile = fopen("destination.txt", "w");

if (destinationFile == NULL) {

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

fclose(sourceFile);

return 1;

}

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

if (fputc(ch, destinationFile) == EOF) {

printf("Error writing to destination file.\n");

fclose(sourceFile);

fclose(destinationFile);

return 1;

}

}

if (fclose(sourceFile) != 0) {

printf("Error closing source file.\n");

return 1;

}

if (fclose(destinationFile) != 0) {

printf("Error closing destination file.\n");

return 1;

}

printf("File copied successfully.\n");

return 0;}

**Question 127: Write a code to show how we can redirect the output of a program, from the screen to a file.**

**Sol.**

#include <stdio.h>

int main() {

FILE \*file;

int i;

file = fopen("output.txt", "w");

if (file == NULL) {

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

return 1;

}

printf("Redirecting output to file...\n");

fprintf(file, "This output is redirected to a file:\n");

for (i = 1; i <= 10; i++) {

printf("Number: %d\n", i);

fprintf(file, "Number: %d\n", i);

}

fclose(file);

printf("Output redirection complete.\n");

return 0;

}

**Question 128: Answer the following questions:**

**(a) How will you use the program given below to perform the following operations?**

* **Copy the contents of one file into another.**
* **Create a new file and add some text to it.**
* **Display the contents of an existing file.**

**# include <stdio.h>**

**int main( )**

**{**

**char ch, str[ 10 ] ;**

**while ( ( ch = fgetc ( stdin ) ) != -1 )**

**fputc ( ch, stdout ) ;**

**return 0 ;**

**}**

**(b) State True or False:**

**1. We can send arguments at command-line even if we define main( ) function without parameters.**

**2. To use standard file pointers we don’t need to open the file using fopen( ).**

**3. The zeroth element of argv array points to the name of the executable file.**

**(c) 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 <old word> <new word> <filename>**

**(d) Write a program that can be used at command prompt as a calculating utility. The usage of the program is shown below.**

**C> calc <switch> <n> <m> 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.**

**Sol.** (a) $ ./a.out < source\_file > target\_file

$ ./a.out > new\_file

$ ./a.out < existing\_file

(b) False

True

True

(c) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

void replace\_word\_in\_file(const char \*old\_word, const char \*new\_word, const char \*filename) {

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

if (!file) {

perror("Error opening file");

exit(EXIT\_FAILURE);

}

FILE \*temp = tmpfile();

if (!temp) {

perror("Error creating temporary file");

fclose(file);

exit(EXIT\_FAILURE);

}

char buffer[1024];

while (fgets(buffer, sizeof(buffer), file)) {

char \*pos;

while ((pos = strstr(buffer, old\_word)) != NULL) {

\*pos = '\0';

fprintf(temp, "%s%s", buffer, new\_word);

strcpy(buffer, pos + strlen(old\_word));

}

fprintf(temp, "%s", buffer);

}

fclose(file);

file = fopen(filename, "w");

if (!file) {

perror("Error reopening file");

fclose(temp);

exit(EXIT\_FAILURE);

}

rewind(temp);

while (fgets(buffer, sizeof(buffer), temp)) {

fputs(buffer, file);

}

fclose(file);

fclose(temp);

}

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

if (argc != 4) {

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

return EXIT\_FAILURE;

}

replace\_word\_in\_file(argv[1], argv[2], argv[3]);

return EXIT\_SUCCESS;

}

(d) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

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

if (argc != 4) {

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

return EXIT\_FAILURE;

}

char \*operator = argv[1];

int n = atoi(argv[2]);

int m = atoi(argv[3]);

if (strcmp(operator, "+") == 0) {

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

} else if (strcmp(operator, "-") == 0) {

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

} else if (strcmp(operator, "\*") == 0) {

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

} else if (strcmp(operator, "/") == 0) {

if (m == 0) {

fprintf(stderr, "Error: Division by zero\n");

return EXIT\_FAILURE;

}

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

} else if (strcmp(operator, "==") == 0) {

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

} else if (strcmp(operator, "!=") == 0) {

printf("%s\n", (n != m) ? "True" : "False");

} else if (strcmp(operator, "<") == 0) {

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

} else if (strcmp(operator, ">") == 0) {

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

} else if (strcmp(operator, "<=") == 0) {

printf("%s\n", (n <= m) ? "True" : "False");

} else if (strcmp(operator, ">=") == 0) {

printf("%s\n", (n >= m) ? "True" : "False");

} else {

fprintf(stderr, "Invalid operator\n");

return EXIT\_FAILURE;

}

return EXIT\_SUCCESS;

}

**Question 129: Write a program that demonstrates the use of >> and << operators:**

**Sol.** #include <stdio.h>

int main() {

unsigned int num1 = 5; // 0000 0101 in binary

unsigned int result1 = num1 << 2; // Shift left by 2 bits

printf("5 << 2 = %u\n", result1); // Expected output: 20 (0001 0100)

unsigned int num2 = 20; // 0001 0100 in binary

unsigned int result2 = num2 >> 2; // Shift right by 2 bits

printf("20 >> 2 = %u\n", result2); // Expected output: 5 (0000 0101)

return 0;

}

**Question 130: Write a program that puts into action both the uses of & operator:**

**Sol.** #include <stdio.h>

int main() {

// Bitwise AND (&) operator

int num1 = 12; // 1100 in binary

int num2 = 6; // 0110 in binary

int result1 = num1 & num2; // Bitwise AND operation

printf("Bitwise AND of %d and %d is %d\n", num1, num2, result1); // Expected output: 4 (0100 in binary)

// Logical AND (&&) operator

int x = 5;

int y = 10;

if (x > 0 && y < 15) { // Logical AND operation

printf("%d is greater than 0 and %d is less than 15\n", x, y);

} else {

printf("%d is not greater than 0 or %d is not less than 15\n", x, y);

}

return 0;

}

**Question 131: 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.**

**Sol.**

#include <stdio.h>

int main() {

unsigned char color;

int number;

// Prompt the user to enter a number

printf("Enter a number: ");

scanf("%d", &number);

// Set the color variable to the input number

color = (unsigned char)number;

// Check each bit representing a color in the rainbow

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;

}

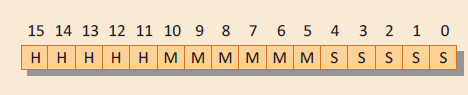
Output:

Enter a number: 41

Blue

Red

**Question 132: The time field in a structure is 2 bytes long. Distribution of different bits which account for hours, minutes and seconds is given in Figure 21.6. Define a function that would receive the 2-byte time and print the equivalent hours, minutes and seconds.**

****

**Sol.** # include <stdio.h>

void display ( unsigned short int time ) ;

int main( )

{

unsigned short int time ;

puts ( "Enter any number less than 24446: " ) ;

scanf ( "%hu", &time ) ;

display ( time ) ;

return 0 ;

}

void display ( unsigned short int tm )

{

unsigned short int hours, minutes, seconds, temp ;

hours = tm >> 11 ;

temp = tm << 5 ;

minutes = temp >> 10 ;

temp = tm << 11 ;

seconds = ( temp >> 11 ) \* 2 ;

printf ( "For Time = %hu\n", tm ) ;

printf ( "Hours = %hu\n", hours ) ;

printf ( "Minutes = %hu\n", minutes ) ;

printf ( "Seconds = %hu\n", seconds ) ;

}

Output:

Enter any number less than 24446:

15500

For Time = 15500

Hours = 7

Minutes = 36

Seconds = 24

**Question 133: In an inter-college competition, various sports like cricket, basketball, football, hockey, lawn tennis, table tennis, carom and chess are played between different colleges. The information regarding the games won by a particular college is stored in bit numbers 0, 1, 2, 3, 4, 5, 6, 7 and 8 of an integer variable game. The college that wins in 5 or more than 5 games is awarded the**

**Champion of Champions trophy. If a number representing the bit pattern mentioned above is entered through the keyboard, then write a program to find out whether the college won the Champion of the Champions trophy or not, along with the names of the games won by the college.**

**Sol.**

#include <stdio.h>

#include <string.h>

void checkChampion(int game)

{

int count = 0;

const char \*games[] = {"Cricket", "Basketball", "Football", "Hockey",

"Lawn Tennis", "Table Tennis", "Carom", "Chess"};

printf("Games won by the college:\n");

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

if (game & (1 << i)) {

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

count++;

}

}

if (count >= 5) {

printf("The college wins the Champion of Champions trophy!\n");

} else {

printf("The college does not win the Champion of Champions trophy.\n");

}

}

int main() {

int game;

printf("Enter the bit pattern representing the games won: ");

scanf("%d", &game);

checkChampion(game);

return 0;

}

**Question 134: An animal could be a canine (dog, wolf, fox, etc.), a feline (cat, lynx, jaguar, etc.), a cetacean (whale, narwhal, etc.) or a marsupial (koala, wombat, etc.). The information whether a particular animal is canine, feline, cetacean, or marsupial is stored in bit number 0, 1, 2 and 3, respectively of an integer variable type. Bit number 4 of the variable type stores the information about whether the animal is Carnivore or Herbivore.**

**For the following animal, complete the program to determine whether the animal is an herbivore or a carnivore. Also determine whether the animal is a canine, feline, cetacean or a marsupial.**

**struct animal**

**{**

**char name[ 30 ] ; int type ;**

**}**

**struct animal a = { "OCELOT", 18 } ;**

**Sol.** #include <stdio.h>

struct animal {

char name[30];

int type;

};

int main() {

struct animal a = { "OCELOT", 18 };

const int CANINE = 1 << 0; // Bit 0

const int FELINE = 1 << 1; // Bit 1

const int CETACEAN = 1 << 2; // Bit 2

const int MARSUPIAL = 1 << 3; // Bit 3

const int CARNIVORE = 1 << 4; // Bit 4

if (a.type & CARNIVORE) {

printf("%s is a carnivore.\n", a.name);

} else {

printf("%s is a herbivore.\n", a.name);

}

if (a.type & CANINE) {

printf("%s is a canine.\n", a.name);

}

if (a.type & FELINE) {

printf("%s is a feline.\n", a.name);

}

if (a.type & CETACEAN) {

printf("%s is a cetacean.\n", a.name);

}

if (a.type & MARSUPIAL) {

printf("%s is a marsupial.\n", a.name);

}

return 0;

}

**Question 135: In order to save disk space, information about student is stored in an integer variable. Bit numbers 0 to 3 indicate whether the student is a Ist year, IInd year, IIIrd year or IVth year student respectively. Bits 4 to 7 indicate whether the student's stream is Mechanical, Chemical, Electronics or CS. Rest of the bits store room number. Such data for 4 students is stored in the following array: int data[ ] = { 273, 548, 786, 1096 } ; Write a program that uses this data and displays the information about the student.**

**Sol.**

#include <stdio.h>

void displayStudentInfo(int data[], int numStudents)

{

const char \*yearNames[] = {"Ist year", "IInd year", "IIIrd year", "IVth year"};

const char \*streamNames[] = {"Mechanical", "Chemical", "Electronics", "CS"};

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

int year = data[i] & 0xF; // Bits 0-3 represent year

int stream = (data[i] >> 4) & 0xF; // Bits 4-7 represent stream

int room = data[i] >> 8; // Rest of the bits represent room number

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

printf("Year: %s\n", yearNames[year]);

printf("Stream: %s\n", streamNames[stream]);

printf("Room number: %d\n", room);

printf("\n");

}

}

int main() {

int data[] = { 273, 548, 786, 1096 };

int numStudents = sizeof(data) / sizeof(data[0]);

displayStudentInfo(data, numStudents);

return 0;

}

**Question 136: What will be the output of the following program?**

**# include <stdio.h>**

**int main( )**

**{**

**int i = 32, j = 65, k, l, m, n, o, p ;**

**k = i | 35 ; l = ~k ; m = i & j ;**

**n = j ^ 32 ; o = j << 2 ; p = i >> 5 ;**

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

**printf ( "n = %d o = %d p = %d\n", n, o, p ) ;**

**return 0 ;**

**}**

**Sol** :

k = 35 l = -36 m = 0

n = 97 o = 260 p = 1

**Question 137: What is hexadecimal equivalent of each of the following binary numbers?**

**01011010 11000011**

**1010101001110101 1111000001011010**

**Sol.** #include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* binaryToHex(char\* binary) {

// Lookup table for hexadecimal digits

char\* hexTable[16] = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F"};

int binaryLength = strlen(binary);

int paddedLength = (binaryLength % 4 == 0) ? binaryLength : (binaryLength + 4 - (binaryLength % 4));

char\* paddedBinary = (char\*)malloc(paddedLength + 1);

memset(paddedBinary, '0', paddedLength);

paddedBinary[paddedLength] = '\0';

strncpy(paddedBinary + (paddedLength - binaryLength), binary, binaryLength);

int hexLength = paddedLength / 4;

char\* hex = (char\*)malloc(hexLength + 1);

hex[hexLength] = '\0';

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

char group[5];

strncpy(group, paddedBinary + i \* 4, 4);

group[4] = '\0';

int decimal = strtol(group, NULL, 2);

hex[i] = \*hexTable[decimal];

}

free(paddedBinary);

return hex;

}

int main() {

char binary1[] = "01011010";

char binary2[] = "11000011";

char binary3[] = "1010101001110101";

char binary4[] = "1111000001011010";

char\* hex1 = binaryToHex(binary1);

char\* hex2 = binaryToHex(binary2);

char\* hex3 = binaryToHex(binary3);

char\* hex4 = binaryToHex(binary4);

printf("Binary: %s, Hexadecimal: %s\n", binary1, hex1);

printf("Binary: %s, Hexadecimal: %s\n", binary2, hex2);

printf("Binary: %s, Hexadecimal: %s\n", binary3, hex3);

printf("Binary: %s, Hexadecimal: %s\n", binary4, hex4);

free(hex1);

free(hex2);

free(hex3);

free(hex4);

return 0;

}

**Question 138: Rewrite the following expressions using bitwise compound assignment operators:**

**a = a | 3 a = a & 0x48 b = b ^ 0x22 c = c << 2**

**Sol.**

a |= 3;

b &= 0x48;

c ^= 0x22;

c <<= 2;

**Question 139: Consider an unsigned integer in which rightmost bit is numbered as 0. Write a function checkbits ( x, p, n ) which returns true if all "n" bits starting from position "p" are turned on, false otherwise. For example, checkbits ( x, 4, 3 ) will return true if bits 4, 3 and 2 are 1 in number x.**

**Sol.**

#include <stdio.h>

#include <stdbool.h>

bool checkbits(unsigned int x, int p, int n) {

unsigned int mask = ((1U << n) - 1) << (p - n + 1);

return (x & mask) == mask;

}

int main() {

unsigned int x = 0xFFFF; // Example number where all bits are set

int p = 4;

int n = 3;

if (checkbits(x, p, n)) {

printf("All %d bits starting from position %d are turned on in %u.\n", n, p, x);

} else {

printf("Not all %d bits starting from position %d are turned on in %u.\n", n, p, x);

}

return 0;

}

**Question 140: Write a program to scan an 8-bit number into a variable and check whether its 3rd, 6th and 7th bit is on.**

**Sol.**

#include <stdio.h>

#include <stdbool.h>

bool checkBits(unsigned char num) {

unsigned char mask = (1 << 2) | (1 << 5) | (1 << 6);

return (num & mask) == mask;

}

int main() {

unsigned char num;

printf("Enter an 8-bit number (0-255): ");

scanf("%hhu", &num);

if (checkBits(num)) {

printf("The 3rd, 6th, and 7th bits are all on.\n");

} else {

printf("The 3rd, 6th, and/or 7th bits are not all on.\n");

}

return 0;

}

**Question 141: Write a program to receive an unsigned 16-bit integer and then exchange the contents of its 2 bytes using bitwise operators.**

**Sol.**

#include <stdio.h>

unsigned short exchangeBytes(unsigned short num) {

unsigned short lowerByte = (num & 0x00FF) << 8;

unsigned short higherByte = (num & 0xFF00) >> 8;

return lowerByte | higherByte;

}

int main() {

unsigned short num;

printf("Enter a 16-bit unsigned number (0-65535): ");

scanf("%hu", &num);

unsigned short exchangedNum = exchangeBytes(num);

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

printf("Exchanged bytes: %hu\n", exchangedNum);

return 0;

}

**Question 142: Write a program to receive an 8-bit number into a variable and then exchange its higher 4 bits with lower 4 bits.**

**Sol.**

#include <stdio.h>

unsigned char exchangeNibbles(unsigned char num) {

unsigned char lowerNibble = (num & 0x0F) << 4;

unsigned char higherNibble = (num & 0xF0) >> 4;

return lowerNibble | higherNibble;

}

int main() {

unsigned char num;

printf("Enter an 8-bit unsigned number (0-255): ");

scanf("%hhu", &num);

unsigned char exchangedNum = exchangeNibbles(num);

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

printf("Exchanged nibbles: %u\n", exchangedNum);

return 0;

}**Question 143: Write a program to receive an 8-bit number into a variable and then set its odd bits to 1.**

**Sol.**

#include <stdio.h>

unsigned char setOddBits(unsigned char num) {

unsigned char mask = 0xAA; // 10101010 in binary

return num | mask;

}

int main() {

unsigned char num;

printf("Enter an 8-bit unsigned number (0-255): ");

scanf("%hhu", &num);

unsigned char result = setOddBits(num);

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

printf("Number with odd bits set to 1: %u\n", result);

return 0;

}

**Question 144: 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.**

**Sol.**

#include <stdio.h>

unsigned char checkAndClearBits(unsigned char num) {

// Check if 3rd and 5th bits are on

if ((num & (1 << 2)) && (num & (1 << 4))) {

// Turn off the 3rd and 5th bits

num &= ~(1 << 2);

num &= ~(1 << 4);

}

return num;

}

int main() {

unsigned char num;

printf("Enter an 8-bit unsigned number (0-255): ");

scanf("%hhu", &num);

unsigned char result = checkAndClearBits(num);

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

printf("Number after checking and clearing 3rd and 5th bits: %u\n", result);

return 0;

}

**Question 145: 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.**

**Sol.**

#include <stdio.h>

unsigned char checkAndSetBits(unsigned char num) {

// Check if 3rd and 5th bits are off

if (!((num & (1 << 2)) || (num & (1 << 4)))) {

// Turn on the 3rd and 5th bits

num |= (1 << 2);

num |= (1 << 4);

}

return num;

}

int main() {

unsigned char num;

printf("Enter an 8-bit unsigned number (0-255): ");

scanf("%hhu", &num);

unsigned char result = checkAndSetBits(num);

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

printf("Number after checking and setting 3rd and 5th bits: %u\n", result);

return 0;

}

**Question 146: Rewrite the showbits( ) function used in this chapter using the \_BV macro.**

**Sol.**

#include <stdio.h>

#define \_BV(bit) (1 << (bit))

void showbits(unsigned char num) {

int i;

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

printf("%d", (num & \_BV(i)) ? 1 : 0);

}

printf("\n");

}

int main() {

unsigned char num;

printf("Enter an 8-bit unsigned number (0-255): ");

scanf("%hhu", &num);

printf("Binary representation of %u: ", num);

showbits(num);

return 0;

}

**Question 147: Write a code to force the compiler to explicitly convert the value of an expression to a particular data type.**

**Sol.**

#include <stdio.h>

int main() {

double num1 = 10.5;

double num2 = 20.7;

int result;

// Force the compiler to convert the result of division to an integer using a type cast

result = (int)(num1 / num2);

printf("Result of division (as integer): %d\n", result);

return 0;

}

Output:

Result of division (as integer): 0

**Question 148: 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.**

**Sol.**

#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 (\*func\_ptr[3])(int, int);

func\_ptr[0] = fun1;

func\_ptr[1] = fun2;

func\_ptr[2] = fun3;

int num1 = 10, num2 = 5;

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

printf("Result of function %d: %.2f\n", i + 1, func\_ptr[i](num1, num2));

}

return 0;

}

Output:

Result of function 1: 15.00

Result of function 2: 5.00

Result of function 3: 50.00

**Question 149: 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.**

**Sol.**

#include <stdio.h>

#include <stdarg.h>

average(int count, ...) {

va\_list args;

double sum = 0.0;

int i;

va\_start(args, count);

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

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

}

va\_end(args);

return sum / count;

}

int main() {

// Calculate the average of different number of arguments

printf("Average of 3, 4, and 5: %.2f\n", find\_average(3, 3.0, 4.0, 5.0));

printf("Average of 2 and 6: %.2f\n", find\_average(2, 2.0, 6.0));

printf("Average of 7, 8, 9, and 10: %.2f\n", find\_average(4, 7.0, 8.0, 9.0, 10.0));

return 0;

}

Output:

Average of 3, 4, and 5: 4.00

Average of 2 and 6: 4.00

Average of 7, 8, 9, and 10: 8.50

**Question 150: What will be the output of the following programs?**

**(a) # include <stdio.h>**

**int main( )**

**{**

**enum status { pass, fail, atkt } ;**

**enum status stud1, stud2, stud3 ;**

**stud1 = pass ;**

**stud2 = fail ;**

**stud3 = atkt ;**

**printf ( "%d %d %d\n", stud1, stud2, stud3 ) ;**

**return 0 ;**

**}**

**(b) # include <stdio.h>**

**int main( )**

**{**

**printf ( "%f\n", ( float ) ( ( int ) 3.5 / 2 ) ) ;**

**printf ( "%d\n", ( int ) ( ( ( float ) 3 / 2 ) \* 3 ) ) ;**

**return 0 ;**

**}**

**Sol.**

(a) 0 1 2

(b) 1.000000

4