KPIT TECHNOLOGIES

WEEKLY REPORT

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

Student name	Week	Branch	USN
Juweriah Herial	3	Circuit (ECE)	1NH20EC053

Yashavant Kanetkar Book 19th edition

Question 101-150:

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.

(a) char str1[10], str2[10], str3[10], str4[10];

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

Scanf will contain

str1 = "Alice"

str2 = "in"

str3 = "wonder"

str4 = "land"

(b)
```c
#include<stdio.h>
#include<conio.h>
int main()
{
	char isbn[15];
	int i, sum = 0;
	printf("\nEnter 10 digit ISBN number : ");
	gets_s(isbn);
	for (i = 0; i <= 9; i++)
	{
		isbn[i] -= 48;
		sum = sum + ((i + 1)*isbn[i];
	}
	if (sum % 11)
		puts("\nISBN number is wrong.");
	else
		puts("\nISBN number is valid.");
	_getch();
	return 0;
}
```

c)
```c
#include<stdio.h>
#include<conio.h>
int main()
{
	char num[20];
	int i, sum = 0;
```

```
        printf("\nEnter the 16 digit credit card number : ");
        scanf("%s", num);

        for (i = 0; i <= 15; i++)
        {
                num[i] -= 48;
                if ((i % 2))
                        sum = sum + num[i];
                else
                {
                        num[i] *= 2;
                        if (num[i] >= 10)
                                num[i] -= 9;
                        sum = sum + num[i];
                }
        }
        if (!(sum % 10))
                printf("\nNumber is valid.");
        else
                printf("\nNumber is not valid.");
        _getch();
        return 0;
}
```

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.

include <stdio.h>

include <string.h>

int main( )

{

char list[6][20] = {

"akshay", "parag", "raman",

"srinivas", "gopal", "rajesh"

} ;

```
int i ;
char yourname[ 20 ] ;
printf ( "Enter your name " ) ;
scanf ( "%s", yourname ) ;
for ( i = 0 ; i <= 5 ; i++ )
{
if ( strcmp ( &list[ i ][ 0 ], yourname ) == 0 )
{
printf ( "Welcome, you can enter the palace\n" ) ;
return 0 ;
}
}
printf ( "Sorry, you are a trespasser" ) ;
return 0 ;
}
```

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.

```
# include <stdio.h>
# include <string.h>
int main( )
{
char *str[ ] = {
"We will teach you how to...",
"Move a mountain", "Level a building",
"Erase the past", "Make a million",
"...all through C!"
} ;
char str1[ 20 ], *p ;
int i ;
printf ( "\nEnter string to be searched: " ) ;
scanf ( "%s", str1 ) ;
p = NULL ;
for ( i = 0 ; i < 6 ; i++ )
```

```
{
p = strstr ( str[ i ], str1 ) ;
if ( p != NULL )
{
printf ( "%s found in the array", str1 ) ;
return 0 ;
}
}
printf ( "%s not found in the array", str1 ) ;
return 0 ;
}
```

Output

```
Enter string to be searched: Million
Million not found in the array
```

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

```
# include <stdio.h>
# include <string.h>
int main( )
{
char *str[ ] = {
"Rajesh", "Ashish", "Milind",
"Pushkar", "Akash"
} ;
char *t ;
int i, j ;
for ( i = 0 ; i < 5 ; i++ )
{
for ( j = i + 1 ; j < 5 ; j++ )
{
if ( ( strcmp ( str[ i ], str[ j ] ) ) > 0 )
{
t = str[ i ] ; str[ i ] = str[ j ] ; str[ j ] = t ;
}
}
}
for ( i = 0 ; i < 5 ; i++ )
printf ( "%s\t", str[ i ] ) ;
return 0 ;
}
```

Output:

```
Akash Ashish Milind Pushkar Rajesh
```

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

```
# include <stdio.h>
# include <string.h>
void xstrrev ( char *ss ) ;
int main( )
{
char str[ ][ 35 ] = {
"To ere is human...",
"But to really mess things up...",
"One needs to know C !!"
} ;
int i ;
for ( i = 0 ; i <= 2 ; i++ )
{
xstrrev ( str[ i ] ) ;
printf ( "%s\n", str[ i ] ) ;
}
return 0 ;
}
void xstrrev ( char *s )
{
int l, i ;
char *t, temp ;
l = strlen ( s ) ;
t = s + l - 1 ;
for ( i = 1 ; i <= l / 2 ; i++ )
{
temp = *s ; *s = *t ; *t = temp ;
s++ ; t-- ;
}
}
```

Output

```
...namuh si ere oT
...pu sgniht ssem yllaer ot tuB
!! C wonk ot sdeen enO
```

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.

(a) char *mess[] = {

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

"Spit and polish", "You and C"

};

(b)
```
#include<stdio.h>
#include<conio.h>
#include<Windows.h>
#define Vowel line[i] == 'A' || line[i] == 'a' || line[i] == 'E' || line[i] == 'e' \
 || line[i] == 'I' || line[i] == 'i' || line[i] == 'O' || line[i] == 'o' || \
 line[i] == 'U' || line[i] == 'u'
```

```
void del_vow(char *line)
{
        int i, j;
        for (i = 0; line[i] != '\0'; i++)
                if (Vowel)
                {
                    for (j = i; line[j] != '\0'; j++)
                                line[j] = line[j + 1];
                        i--;
                }

}
int main()
{
        char line[80];
        puts("Enter the line");
        gets_s(line);
        del_vow(line);
        printf("\nLine without vowels\n");
        puts(line);
        getch();
        return 0;
}
```

```
(c #include<stdio.h>
#include<conio.h>
#include<string.h>

#define Space 32
#define (line[i] == 't' || line[i] == 'T') && (line[i + 1] == 'h' || \
line[i + 1] == 'H') && (line[i + 2] == 'E' || line[i + 2] == 'e') && \
(line[i + 3] == Space || line[i + 3] == '\0'

void del_the(char *line)
{
        int i, j;
        for (i = 0; line[i] != '\0'; i++)
```

```
                if ()/*if The is encounter*/
                        for (j = i; line[j] != '\0'; j++)
                                line[j] = line[j + 4];
}
int main()
{
        char line[80];
        puts("Enter the line");
        gets_s(line);
        del_the(line);
        puts("\nAfter removing all the words 'The'.\n\n");
        puts(line);
        _getch();
        return 0;
}
```

d)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<Windows.h>
#include<malloc.h>

#define Space 32

char* last_name(char *line)
{
        char temp[20], *p;
        int i, j, l = 0;
        for (i = j = 0; line[i] != '\0'; i++)
        {
                if (line[i] == Space)
                {
                        temp[l] = line[j];
                        l++;
                        temp[l] = Space;
                        l++;
                        j = i + 1;
```

```
                }
        }
        if (line[i] == '\0')
        {
                for (; line[j] != '\0'; j++, l++)
                        temp[l] = line[j];
                temp[l] = line[j];
        }

        p = (char*)malloc(sizeof(strlen(temp) + 1));
        strcpy(p, temp);
        return p;
}

int main()
{
        char *name[10];
        int i = 0, j;
        char ans = 'y', *p, naam[30];
        while (ans == 'y')
        {
                puts("\nEnter the full name : ");
                gets_s(naam);
                p = (char*)malloc(sizeof(strlen(naam) + 1));
                strcpy(p, naam);
                name[i] = p;
                name[i] = last_name(name[i]);
                printf("\nWant to enter another name (y/n) : ");
                scanf("%c", &ans);
                i++;
                while (getchar() != '\n'
                if (i > 9)
                        break;
        }
        if (i >= 10)
                puts("\nNo more names can be entered");
        system("cls");
        puts("\n\t\tName in the given format.");
```

```
        for (j = 0; j < i; j++)
                puts(name[j]);
        _getch();
        return 0;
}
```

(e)

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int isVowel(char ch) {
    char vowels[] = "AEIOUaeiou";
    for (int i = 0; vowels[i] != '\0'; i++) {
        if (ch == vowels[i]) {
            return 1;
        }
    }
    return 0;
}

int countVowelPairs(const char *str) {
    int count = 0;
    for (int i = 0; str[i] != '\0'; i++) {
        if (isVowel(str[i]) && isVowel(str[i + 1])) {
            count++;
        }
    }
    return count;
}

int main() {
    char sentence[100];
    printf("Enter a sentence: ");
    fgets(sentence, sizeof(sentence), stdin);
    int count = countVowelPairs(sentence);
    printf("Number of occurrences of two vowels in succession: %d\n", count);
    return 0;
}
```

(f)

```
#include <stdio.h>
#include <string.h>

void printNumberInWords(int num);
```

```c
const char *one[] = { "", "One ", "Two ", "Three ", "Four ", "Five ", "Six ",
"Seven ", "Eight ", "Nine " };
const char *ten[] = { "", "", "Twenty ", "Thirty ", "Forty ", "Fifty ", "Sixty
", "Seventy ", "Eighty ", "Ninety " };
const char *eleven_to_nineteen[] = { "Ten ", "Eleven ", "Twelve ", "Thirteen
", "Fourteen ", "Fifteen ", "Sixteen ", "Seventeen ", "Eighteen ", "Nineteen "
};

void convert_to_words(int num, char *result) {
    if (num >= 1000) {
        strcat(result, one[num / 1000]);
        strcat(result, "Thousand ");
        num %= 1000;
    }
    if (num >= 100) {
        strcat(result, one[num / 100]);
        strcat(result, "Hundred ");
        num %= 100;
    }
    if (num >= 20) {
        strcat(result, ten[num / 10]);
        strcat(result, one[num % 10]);
    } else if (num >= 10) {
        strcat(result, eleven_to_nineteen[num % 10]);
    } else {
        strcat(result, one[num]);
    }
}

void printNumberInWords(int num) {
    char result[500] = "";
    if (num == 0) {
        strcpy(result, "Zero");
    } else {
        convert_to_words(num, result);
    }
    printf("%s\n", result);
}

int main() {
    int num;
    printf("Enter an integer (<= 9 digits): ");
    scanf("%d", &num);
    printNumberInWords(num);
    return 0;
}
```

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.

```
# include <stdio.h>

int main( )
{
char name[ 3 ] ;
float price[ 3 ] ;
int pages[ 3 ], i ;
printf ( "Enter names, prices and no. of pages of 3 books\n" ) ;
for ( i = 0 ; i <= 2 ; i++ )
scanf ( "%c %f %d", &name[ i ], &price[ i ], &pages[ i ] ) ;
printf ( "And this is what you entered\n" ) ;
for ( i = 0 ; i <= 2 ; i++ )
printf ( "%c %f %d\n", name[ i ], price[ i ], pages[ i ] ) ;
return 0 ;
}
```

And here is the sample run...

```
Enter names, prices and no. of pages of 3 books
A 100.00 354
C 256.50 682
F 233.70 512
And this is what you entered
A 100.000000 354
C 256.500000 682
F 233.700000 512
```

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.

```
# include <stdlib.h>
# include <stdio.h>
struct node
{
int data ; struct node *link ;
} ;
void push ( struct node **s, int item ) ;
int pop ( struct node **s ) ;
int main( )
```

```
{
struct node *top ;
int t, i, item ;
top = NULL ;
push ( &top, 45 ) ; push ( &top, 28 ) ;
push ( &top, 63 ) ; push ( &top, 55 ) ;
item = pop ( &top ) ;
printf ( "Popped : %d\n", item ) ;
item = pop ( &top ) ;
printf ( "Popped : %d\n", item ) ;
return 0 ;
}
void push ( struct node **s, int item )
{
struct node *q ;
q = ( struct node * ) malloc ( sizeof ( struct node ) ) ;
q -> data = item ;
q -> link = *s ;
*s = q ;
}
int pop ( struct node **s )
{
int item ;
struct node *q ;
if ( *s == NULL )
printf ( "Stack is empty\n" ) ;
else
{
q = *s ;
item = q -> data ;
*s = q -> link ;
free ( q ) ;
return ( item ) ;
}
}
```

Output

Popped : 55

Popped : 63

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.

```
# include <stdio.h>
# include <stdlib.h>
struct queue
{
int item ; struct queue *link ;
} ;
struct queue *rear, *front ;
void add ( int item ) ;
int del_queue( ) ;
int main( )
{
int item ;
rear = front = NULL ;
add ( 10 ) ; add ( 20 ) ; add ( 30 ) ;
add ( 40 ) ; add ( 50 ) ; add ( 60 ) ;
item = del_queue( ) ;
printf ( "Deleted Item = %d\n", item ) ;
item = del_queue( ) ;
printf ( "Deleted Item = %d\n", item ) ;
return 0 ;
}
void add ( int item )
{
struct queue *q = ( struct queue * ) malloc ( sizeof ( struct queue ) ) ;
q -> item = item ;
q -> link = NULL ;
if ( rear == NULL )
{
rear = q ; front = q ;
}
else
{
q -> link = rear ; rear = q ;
}
}
int del_queue( )
{
int item ;
struct queue *q = rear ;
if ( front == NULL )
{
```

```
printf ( "Queue is empty\n" ) ;
return -1;
}
else
{
if ( front == rear )
{
item = q -> item ; front = rear = NULL ;
free( q ) ;
}
else
{
while( q -> link -> link != NULL )
q = q -> link ;
item = q -> link -> item ;
free( q -> link ) ;
front = q ;
q -> link = NULL ;
}
}
return item ;
}
```

Output:
Deleted Item = 10
Deleted Item = 20

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

(a) maruti.engine.bolts = 25;

(b) struct time {

    int hours;

    int minutes;

    int seconds;

} t;

struct time *pt;

pt = &t;

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

(a)
```c
#include <stdio.h>

#include <string.h>
#define MAX_STUDENTS 450
struct student {
    int roll_number;
    char name[50];
    char department[30];
    char course[30];
    int year_of_joining;
};
void print_students_by_year(struct student students[], int n, int year) {
    printf("Students who joined in %d:\n", year);
    for (int i = 0; i < n; i++) {
        if (students[i].year_of_joining == year) {
            printf("%s\n", students[i].name);
        }
    }
}
void print_student_by_roll_number(struct student students[], int n, int
roll_number) {
    for (int i = 0; i < n; i++) {
        if (students[i].roll_number == roll_number) {
            printf("Roll Number: %d\n", students[i].roll_number);
            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].year_of_joining);
            return;
        }
    }
```

```
    printf("Student with roll number %d not found.\n", roll_number);
}
int main() {
    struct student students[MAX_STUDENTS] = {
        {1, "Alice", "CS", "B.Tech", 2020},
        {2, "Bob", "ECE", "B.Tech", 2021},
        {3, "Charlie", "ME", "B.Tech", 2020},
    };
    int year = 2020;
    print_students_by_year(students, 3, year);
    int roll_number = 2;
    print_student_by_roll_number(students, 3, roll_number);
    return 0;
}
```

(b)

```
#include<stdio.h>
#include<conio.h>
#include<Windows.h>

/*Function to perform withdrawal or deposition*/
void action(int, int, int);

/*Print the balance below 100 Rs.*/
void below100();

struct acc_holder
{
        long int acc_num;
        char name[30];
        int bal;
} sbi[200] = { 1, "Siraj", 1000000,
2, "Azad", 1233044,
3, "Deepak", 99,
4, "Rihan", 33,
5, "Rahul Khowal", 200000
};

int main()
{
        int accnum, amount, code;
```

```
        printf("\nEnter your account number : ");
        scanf("%d", &accnum);
        printf("Enter 1 for deposit and 0 for withdrawal : ");
        scanf("%d", &code);
        if (code)
        {
                printf("\nEnter amount to be deposit : ");
                scanf("%d", &amount);
        }
        else
        {
                printf("\nEnter amount to withdraw : ");
                scanf("%d", &amount);
        }
        action(accnum, amount, code);
        _getch();
        system("cls");
        printf("All members with account balance less than 100 are following :
");
        below100();
        _getch();
        return 0;
}

void below100()
{
        int i;
        for (i = 0; i < 200; i++)
        {
                if (sbi[i].bal < 100 && sbi[i].bal > 0)
                {
                        printf("\nName : %s", sbi[i].name);
                        printf("\nAccount Number : %d\n\n", sbi[i].acc_num);
                }
        }
}

void action(int accnum, int amount, int code)
```

```
{
        int i;
        for (i = 0; i < 200; i++)
                if (sbi[i].acc_num == accnum)
                        break;
        if (!code)
        {
                if (sbi[i].bal - amount < 100)
                {
                        printf("\nThe balance is insufficient for the
specified withdrawal");
                        return;
                }
                else
                {
                        sbi[i].bal -= amount;
                        printf("\nYour new account balance is : %d",
sbi[i].bal);
                }
        }
        else
        {
                sbi[i].bal += amount;
                printf("\nYour new account balance is : %d", sbi[i].bal);
        }

}
```

(c)
```
#include <stdio.h>
#include <string.h>
struct engine_part {
    char serial_number[4];
    int year_of_manufacture;
    char material[20];
    int quantity_manufactured;
};

void retrieve_parts(struct engine_part parts[], int n, const char *start,
const char *end) {
    printf("Parts with serial numbers between %s and %s:\n", start, end);
```

```
    for (int i = 0; i < n; i++) {
        if (strcmp(parts[i].serial_number, start) >= 0 &&
strcmp(parts[i].serial_number, end) <= 0) {
            printf("Serial Number: %s, Year: %d, Material: %s, Quantity:
%d\n",
                   parts[i].serial_number, parts[i].year_of_manufacture,
parts[i].material, parts[i].quantity_manufactured);
        }
    }
}
int main() {
    struct engine_part parts[] = {
        {"AA1", 2021, "Steel", 100},
        {"BB2", 2022, "Aluminum", 200},
        {"CC5", 2021, "Iron", 150},
    };
    retrieve_parts(parts, 3, "BB1", "CC6");
    return 0;
}
```

(d)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct cricketer {
    char name[50];
    int age;
    int matches;
    float average_runs;
};
int compare_cricketers(const void *a, const void *b) {
    struct cricketer *cricketerA = (struct cricketer *)a;
    struct cricketer *cricketerB = (struct cricketer *)b;
    if (cricketerA->average_runs < cricketerB->average_runs)
        return -1;
    else if (cricketerA->average_runs > cricketerB->average_runs)
        return 1;
    else
        return 0;
}
int main() {
    struct cricketer cricketers[20] = {
        {"Sachin", 40, 200, 55.0},
        {"Rahul", 38, 180, 52.5},
        {"Virat", 32, 90, 50.0},
    };
    int n = 3; // Number of cricketers currently in the array
```

```
    qsort(cricketers, n, sizeof(struct cricketer), compare_cricketers);
    printf("Cricketers sorted by average runs:\n");
    for (int i = 0; i < n; i++) {
        printf("Name: %s, Age: %d, Matches: %d, Average Runs: %.2f\n",
               cricketers[i].name, cricketers[i].age, cricketers[i].matches,
cricketers[i].average_runs);
    }
    return 0;
}
```

e)
```
#include <stdio.h>
#include <string.h>
struct date {
    int day;
    int month;
    int year;
};

struct employee {
    int code;
    char name[50];
    struct date date_of_joining;
};
int calculate_years(struct date start, struct date end) {
    int years = end.year - start.year;
    if (end.month < start.month || (end.month == start.month && end.day <
start.day)) {
        years--;
    }
    return years;
}
void print_employees_with_tenure(struct employee employees[], int n, struct
date current_date) {
    printf("Employees with tenure >= 3 years:\n");
    for (int i = 0; i < n; i++) {
        if (calculate_years(employees[i].date_of_joining, current_date) >= 3)
{
            printf("Name: %s\n", employees[i].name);
        }
    }
}
int main() {
    struct employee employees[5] = {
        {1, "Alice", {15, 6, 2018}},
        {2, "Bob", {20, 3, 2019}},
        {3, "Charlie", {10, 11, 2020}},
```

```
    };
    int n = 3;
    struct date current_date;
    printf("Enter current date (dd mm yyyy): ");
    scanf("%d %d %d", &current_date.day, &current_date.month,
&current_date.year);
    print_employees_with_tenure(employees, n, current_date);
    return 0;
}
```

f)

```
#include <stdio.h>
#include <string.h>
#define MAX_BOOKS 100
struct library {
    int accession_number;
    char title[50];
    char author[50];
    float price;
    int is_issued;
};

void add_book(struct library books[], int *count) {
    if (*count >= MAX_BOOKS) {
        printf("Library is full!\n");
        return;
    }
    printf("Enter accession number: ");
    scanf("%d", &books[*count].accession_number);
    printf("Enter title: ");
    scanf(" %[^\n]%*c", books[*count].title);
    printf("Enter author: ");
    scanf(" %[^\n]%*c", books[*count].author);
    printf("Enter price: ");
    scanf("%f", &books[*count].price);
    books[*count].is_issued = 0;
    (*count)++;
}

void display_books(struct library books[], int count) {
    for (int i = 0; i < count; i++) {
        printf("Accession Number: %d\n", books[i].accession_number);
        printf("Title: %s\n", books[i].title);
        printf("Author: %s\n", books[i].author);
        printf("Price: %.2f\n", books[i].price);
        printf("Issued: %s\n", books[i].is_issued ? "Yes" : "No");
        printf("\n");
```

```
    }
}
void list_books_by_author(struct library books[], int count, const char
*author) {
    for (int i = 0; i < count; i++) {
        if (strcmp(books[i].author, author) == 0) {
            printf("Title: %s\n", books[i].title);
        }
    }
}
void list_title_by_accession_number(struct library books[], int count, int
accession_number) {
    for (int i = 0; i < count; i++) {
        if (books[i].accession_number == accession_number) {
            printf("Title: %s\n", books[i].title);
            return;
        }
    }
    printf("Book with accession number %d not found.\n", accession_number);
}
void list_book_count(int count) {
    printf("Total number of books: %d\n", count);
}
void list_books_by_accession_order(struct library books[], int count) {
    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (books[j].accession_number > books[j + 1].accession_number) {
                struct library temp = books[j];
                books[j] = books[j + 1];
                books[j + 1] = temp;
            }
        }
    }
    display_books(books, count);
}
int main() {
    struct library books[MAX_BOOKS];
    int count = 0;
    int choice;
    while (1) {
        printf("Library 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);
        switch (choice) {
            case 1:
                add_book(books, &count);
                break;
            case 2:
                display_books(books, count);
                break;
            case 3: {
                char author[50];
                printf("Enter author name: ");
                scanf(" %[^\n]%*c", author);
                list_books_by_author(books, count, author);
                break;
            }
            case 4: {
                int accession_number;
                printf("Enter accession number: ");
                scanf("%d", &accession_number);
                list_title_by_accession_number(books, count,
accession_number);
                break;
            }
            case 5:
                list_book_count(count);
                break;
            case 6:
                list_books_by_accession_order(books, count);
                break;
            case 7:
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
```

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 ;
}
```

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

If the input character is A, the output will be a.

If the input character is 1, the output will be 1 (since 1 does not change case).

(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.

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 ;
}
```

(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.

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

(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.

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

```
# include <stdio.h>
int main( )
{
FILE *fp ;
char ch ;
fp = fopen ( "PR1.C", "r" ) ;
while ( 1 )
{
ch = fgetc ( fp ) ;
if ( ch == EOF )
break ;
printf ( "%c", ch ) ;
}
printf ( "\n" ) ;
fclose ( fp ) ;
return 0 ;
}
```

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

```
# include <stdio.h>
int main( )
{
FILE *fp ;
char ch ;
int nol = 0, not = 0, nob = 0, noc = 0 ;
fp = fopen ( "PR1.C", "r" ) ;
while ( 1 )
{
ch = fgetc ( fp ) ;
if ( ch == EOF )
break ;
noc++ ;
if ( ch == ' ' )
nob++ ;
if ( ch == '\n' )
nol++ ;
if ( ch == '\t' )
```

```
not++ ;
}
fclose ( fp ) ;
printf ( "Number of characters = %d\n", noc ) ;
printf ( "Number of blanks = %d\n", nob ) ;
printf ( "Number of tabs = %d\n", not ) ;
printf ( "Number of lines = %d\n", nol ) ;
return 0 ;
}
```

OUTPUT:

```
Number of characters = 125
Number of blanks = 25
Number of tabs = 13
Number of lines = 22
```

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

```
# include <stdio.h>
# include <stdlib.h>
int main( )
{
FILE *fs, *ft ;
char ch ;
fs = fopen ( "PR1.C", "r" ) ;
if ( fs == NULL )
{
puts ( "Cannot open source file" ) ; exit ( 1 ) ;
}
ft = fopen ( "PR2.C", "w" ) ;
if ( ft == NULL )
{
puts ( "Cannot open target file" ) ;
fclose ( fs ) ; exit ( 2 ) ;
}
while ( 1 )
{
ch = fgetc ( fs ) ;
if ( ch == EOF )
break ;
```

```
else
fputc ( ch, ft ) ;
}
fclose ( fs ) ; fclose ( ft ) ;
return 0 ;
}
```

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

```
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
int main( )
{
FILE *fp ;
char str[ 80 ] ;
fp = fopen ( "POEM.TXT", "w" ) ;
if ( fp == NULL )
{
puts ( "Cannot open file" ) ; exit ( 1 ) ;
}
printf ( "\nEnter a few lines of text:\n" ) ;
while ( strlen ( gets ( str ) ) > 0 )
{
fputs ( str, fp ) ; fputs ( "\n", fp ) ;
}
fclose ( fp ) ;
printf ( "\nFile contents are being read now…\n" , s ) ;
fp = fopen ( "POEM.TXT", "r" ) ;
if ( fp == NULL )
{
puts ( "Cannot open file" ) ; exit ( 2 ) ;
}
while ( fgets ( str, 79, fp ) != NULL )
printf ( "%s" , str ) ;
fclose ( fp ) ;
return 0 ;
}
```

119) Write a program to record the I/O functions in files.

```
# include <stdio.h>
int main( )
```

```
{
FILE *fp ;
struct emp
{
char name[ 40 ] ; int age ; float bs ;
} ;
struct emp e ;
char ch = 'Y' ;
fp = fopen ( "EMPLOYEE.DAT", "w" ) ;
while ( ch == 'Y' )
{
printf ( "Enter name, age, salary: " ) ;
scanf ( "%s %d %f", e.name, &e.age, &e.bs ) ;
fprintf ( fp, "%s %d %f\n", e.name, e.age, e.bs ) ;
printf ( "Another record: " ) ;
ch = fgetchar( ) ;
}
fclose ( fp ) ;
fp = fopen ( "EMPLOYEE.DAT", "r" ) ;
while ( fscanf ( fp, "%s %d %f", e.name, &e.age, &e.bs ) != EOF )
printf ( "%s %d %f\n", e.name, e.age, e.bs ) ;
fclose ( fp ) ;
ch = 'Y' ;
fp = fopen ( "EMP.DAT", "wb" ) ;
while ( ch == 'Y' )
{
printf ( "Enter name, age, salary: " ) ;
scanf ( "%s %d %f", e.name, &e.age, &e.bs ) ;
fwrite ( &e, sizeof ( e ), 1, fp ) ;
printf ( "Another record: " ) ;
ch = fgetchar( ) ;
}
fclose ( fp ) ;
fp = fopen ( "EMP.DAT", "rb" ) ;
while ( fread ( &e, sizeof ( e ), 1, fp ) == 1 )
printf ( "%s %d %f\n", e.name, e.age, e.bs ) ;
fclose ( fp ) ;
return 0 ;
}
```

120) Write a program that performs all the file operations.

```
# include <fcntl.h>
```

```
# include <sys\types.h>
# include <sys\stat.h>
# include <stdlib.h>
# include <stdio.h>
int main( )
{
char buffer[ 512 ], source[ 128 ], target[ 128 ] ;
int in, out, bytes ;
printf ( "\nEnter source file name: " ) ;
gets ( source ) ;
in = open ( source, O_RDONLY | O_BINARY ) ;
if ( in == -1 )
{
puts ( "Cannot open file" ) ; exit ( 1 ) ;
}
printf ( "\nEnter target file name: " ) ;
gets ( target ) ;
out = open ( target, O_CREAT | O_BINARY | O_WRONLY, S_IWRITE) ;
if ( out == -1 )
{
puts ( "Cannot open file" ) ;
close ( in ) ; exit ( 2 ) ;
}
while ( ( bytes = read ( in, buffer, 512 ) ) > 0 )
write ( out, buffer, bytes ) ;
close ( in ) ; close ( out ) ;
return 0 ;
```

}

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

```
# include <stdio.h>
# include <stdlib.h>
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 ;
}
```

Output:

```
Enter the file name: Sample.txt
1: What is this life
2: if full of care
3: We have no time
4: to stand and stare!
```

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

```
# include <stdio.h>
```

```
# include <stdlib.h>
# include <string.h>
int main( )
{
FILE *fs, *ft ;
char source[ 67 ], target[ 67 ], str[ 80 ] ;
puts ( "Enter source file name: " ) ;
gets ( source ) ;
puts ( "Enter target file name: " ) ;
gets ( target ) ;
fs = fopen ( source, "r" ) ;
if ( fs == NULL )
{
puts ( "Unable to open source file" ) ; exit ( 0 ) ;
}
ft = fopen ( target, "a" ) ;
if ( ft == NULL )
{
fclose ( fs ) ;
puts ( "Unable to open target file" ) ; exit ( 0 ) ;
}
while ( fgets ( str, 79, fs ) != NULL )
fputs ( str, ft ) ;
printf ( "Appending file completed!!" ) ;
fclose ( fs ) ;
fclose ( ft ) ;
return 0 ;
}
```

Output:

```
Enter source file name:
Sample.txt
Enter target file name:
NewSample.txt
Appending file completed!!
```

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?
(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.

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.
(a) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_STUDENTS 100

#define NAME_LEN 30

struct student {

    char name[NAME_LEN];

    int age;

};

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

    struct student *studentA = (struct student *)a;

    struct student *studentB = (struct student *)b;

    return strcmp(studentA->name, studentB->name);

}

int main() {

    struct student students[MAX_STUDENTS];

    int count = 0;

    FILE *file = fopen("students.txt", "r");

```
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }
    while (fscanf(file, "%s %d", students[count].name, &students[count].age) != EOF) {
        count++;
    }
    fclose(file);
    qsort(students, count, sizeof(struct student), compare);
    for (int i = 0; i < count; i++) {
        printf("%s %d\n", students[i].name, students[i].age);
    }
    return EXIT_SUCCESS;
}
```

(b)

```
#include <stdio.h>
#include <ctype.h>
int main() {
    char source[128], target[128];
    FILE *srcFile, *tgtFile;
    int ch;
    printf("Enter source file name: ");
    scanf("%s", source);
    printf("Enter target file name: ");
    scanf("%s", target);
    srcFile = fopen(source, "r");
    if (srcFile == NULL) {
        perror("Error opening source file");
```

```
        return EXIT_FAILURE;
    }
    tgtFile = fopen(target, "w");
    if (tgtFile == NULL) {
        perror("Error opening target file");
        fclose(srcFile);
        return EXIT_FAILURE;
    }
    while ((ch = fgetc(srcFile)) != EOF) {
        fputc(toupper(ch), tgtFile);
    }
    fclose(srcFile);
    fclose(tgtFile);
    return EXIT_SUCCESS;
}
```

(c)

```
#include <stdio.h>
int main() {
    char file1[128], file2[128], target[128];
    FILE *fp1, *fp2, *fpTarget;
    char line1[512], line2[512];
    printf("Enter first file name: ");
    scanf("%s", file1);
    printf("Enter second file name: ");
    scanf("%s", file2);
    printf("Enter target file name: ");
    scanf("%s", target);
    fp1 = fopen(file1, "r");
```

```
    if (fp1 == NULL) {
        perror("Error opening first file");
        return EXIT_FAILURE;
    }
    fp2 = fopen(file2, "r");
    if (fp2 == NULL) {
        perror("Error opening second file");
        fclose(fp1);
        return EXIT_FAILURE;
    }
    fpTarget = fopen(target, "w");
    if (fpTarget == NULL) {
        perror("Error opening target file");
        fclose(fp1);
        fclose(fp2);
        return EXIT_FAILURE;
    }
    while (fgets(line1, sizeof(line1), fp1) != NULL && fgets(line2, sizeof(line2), fp2) != NULL) {
        fputs(line1, fpTarget);
        fputs(line2, fpTarget);
    }
    while (fgets(line1, sizeof(line1), fp1) != NULL) {
        fputs(line1, fpTarget);
    }
    while (fgets(line2, sizeof(line2), fp2) != NULL) {
        fputs(line2, fpTarget);
    }
```

```
    fclose(fp1);
    fclose(fp2);
    fclose(fpTarget);
    return EXIT_SUCCESS;
}
```

(d)

```
#include <stdio.h>
#include <stdlib.h>
void offset_cipher(const char *source, const char *target, int offset) {
    FILE *srcFile, *tgtFile;
    int ch;
    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 ((ch = fgetc(srcFile)) != EOF) {
        fputc(ch + offset, tgtFile);
    }

    fclose(srcFile);
    fclose(tgtFile);
```

```
}
int main() {
    char source[128], target[128];
    int offset = 128;
    printf("Enter source file name: ");
    scanf("%s", source);
    printf("Enter target file name: ");
    scanf("%s", target);
    offset_cipher(source, target, offset);
    return EXIT_SUCCESS;
}
```

(e)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct customer {
    int accno;
    char name[30];
    float balance;
};
struct trans {
    int accno;
    char trans_type;
    float amount;
};
int main() {
    FILE *custFile, *transFile;
    struct customer customers[10];
    struct trans transaction;
```

```
    int i, count = 0;
    custFile = fopen("CUSTOMER.DAT", "rb+");
    if (custFile == NULL) {
        perror("Error opening customer file");
        return EXIT_FAILURE;
    }
    while (fread(&customers[count], sizeof(struct customer), 1, custFile)) {
        count++;
    }
    transFile = fopen("TRANSACTIONS.DAT", "rb");
    if (transFile == NULL) {
        perror("Error opening transaction file");
        fclose(custFile);
        return EXIT_FAILURE;
    }
    while (fread(&transaction, sizeof(struct trans), 1, transFile)) {
        for (i = 0; i < count; i++) {
            if (customers[i].accno == transaction.accno) {
                if (transaction.trans_type == 'D') {
                    customers[i].balance += transaction.amount;
                } else if (transaction.trans_type == 'W') {
                    if (customers[i].balance - transaction.amount >= 100) {
                        customers[i].balance -= transaction.amount;
                    } else {
                        printf("Insufficient balance for account %d\n",
transaction.accno);
                    }
                }
```

```
            }
        }
    }
    fclose(transFile);
    fseek(custFile, 0, SEEK_SET);
    fwrite(customers, sizeof(struct customer), count, custFile);
    fclose(custFile);
    return EXIT_SUCCESS;
}
```

(f)

```
#include <stdio.h>
#include <stdlib.h>
struct date {
    int d, m, y;
};
struct employee {
    int empcode[6];
    char empname[20];
    struct date join_date;
    float salary;
};
int compare_dates(const void *a, const void *b) {
    struct employee *empA = (struct employee *)a;
    struct employee *empB = (struct employee *)b;
    if (empA->join_date.y != empB->join_date.y)
        return empA->join_date.y - empB->join_date.y;
    if (empA->join_date.m != empB->join_date.m)
        return empA->join_date.m - empB->join_date.m;
```

```
    return empA->join_date.d - empB->join_date.d;
}
int main() {
    struct employee employees[10];
    FILE *file = fopen("EMPLOYEES.DAT", "r");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }
    for (int i = 0; i < 10; i++) {
        fread(&employees[i], sizeof(struct employee), 1, file);
    }
    fclose(file);
    qsort(employees, 10, sizeof(struct employee), compare_dates);
    file = fopen("SORTED_EMPLOYEES.DAT", "w");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }
    for (int i = 0; i < 10; i++) {
        fwrite(&employees[i], sizeof(struct employee), 1, file);
    }
    fclose(file);
    return EXIT_SUCCESS;
}
```

(g)

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct donor {
    char name[20];
    char address[40];
    int age;
    int blood_type;
};
int main() {
    struct donor donor;
    FILE *file = fopen("DONORS.DAT", "r");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }
    while (fread(&donor, sizeof(struct donor), 1, file)) {
        if (donor.age < 25 && donor.blood_type == 2) {
            printf("Name: %s\nAddress: %s\nAge: %d\nBlood Type: %d\n",
donor.name, donor.address, donor.age, donor.blood_type);
        }
    }
    fclose(file);
    return EXIT_SUCCESS;
}
```

(h)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char names[100][30];
    int n, i = 0;
    FILE *file;
```

```
file = fopen("students.txt", "w");
if (file == NULL) {
    perror("Error opening file");
    return EXIT_FAILURE;
}
printf("Enter student names (type 'end' to stop):\n");
while (1) {
    scanf("%s", names[i]);
    if (strcmp(names[i], "end") == 0) break;
    fprintf(file, "%s\n", names[i]);
    i++;
}
fclose(file);
printf("Enter the position of the name to display: ");
scanf("%d", &n);
if (n <= 0 || n > i) {
    printf("Invalid position\n");
    return EXIT_FAILURE;
}
file = fopen("students.txt", "r");
if (file == NULL) {
    perror("Error opening file");
    return EXIT_FAILURE;
}
for (int j = 0; j < n; j++) {
    fgets(names[0], 30, file);
}
```

```
    printf("The %dth name is: %s", n, names[0]);
    fclose(file);
    return EXIT_SUCCESS;
}
```

(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;
}
```

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

```
# include <stdio.h>
# include <stdlib.h>
int main ( int argc, char *argv[ ] )
{
FILE *fs, *ft ;
char ch ;
```

```
if ( argc != 3 )
{
puts ( "Improper number of arguments\n" ) ;
exit ( 1 ) ;
}
fs = fopen ( argv[ 1 ], "r" ) ;
if ( fs == NULL )
{
puts ( "Cannot open source file\n" ) ;
exit ( 2 ) ;
}
ft = fopen ( argv[ 2 ], "w" ) ;
if ( ft == NULL )
{
puts ( "Cannot open target file\n" ) ;
fclose ( fs ) ;
exit ( 3 ) ;
}
while ( 1 )
{
ch = fgetc ( fs ) ;
if ( ch == EOF )
break ;
else
fputc ( ch, ft ) ;
}
fclose ( fs ) ;
fclose ( ft ) ;
return 0 ;
}
```

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

```
# include <stdio.h>
int main( )
{
FILE *fp ;
char ch ;
fp = fopen ( "TRIAL", "w" ) ;
while ( !feof ( fp ) )
{
ch = fgetc ( fp ) ;
```

```
if ( ferror( ) )
{
perror ( "TRIAL" ) ;
break ;
}
else
printf ( "%c", ch ) ;
}
fclose ( fp ) ;
return 0 ;
}
```

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

```
C>UTIL.EXE
perhaps I had a wicked childhood,
perhaps I had a miserable youth,
but somewhere in my wicked miserable past,
there must have been a moment of truth ^Z
C>
```

Or

```
C>UTIL.EXE > POEM.TXT
C>
```

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.
(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;
}
```

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

```
# include <stdio.h>
void showbits ( unsigned char ) ;
int main( )
```

```
{
unsigned char num = 225, k ;
printf ( "\nDecimal %d is same as binary ", num ) ;
showbits ( num ) ;
k = num >> 1 ;
printf ( "\n%d right shift 1 gives ", num ) ; showbits ( k ) ;
k = num >> 2 ;
printf ( "\n%d right shift 2 gives ", num ) ; showbits ( k ) ;
k = num << 1 ;
printf ( "\n%d left shift 1 gives ", num ) ; showbits ( k ) ;
k = num << 2 ;
printf ( "\n%d left shift 2 gives ", num ) ; 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" ) ;
}
}
```

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

```
# include <stdio.h>

void showbits ( unsigned char ) ;
int main( )
{
unsigned char num = 0xAD, j ;
printf ( "\nValue of num = " ) ;
showbits ( num ) ;
j = num & 0x20 ;
if ( j == 0 )
printf ( "\nIts fifth bit is off" ) ;
else
printf ( "\nIts fifth bit is on" ) ;
j = num & 0x08 ;
if ( j == 0 )
```

```
printf ( "\nIts third bit is off" ) ;
else
{
printf ( "\nIts third bit is on" ) ;
num = num & 0xF7 ;
printf ( "\nNew value of num = " ) ;
showbits ( num ) ;
j = num & 0x08 ;
if ( j == 0 )
printf ( "\nNow its third bit is turned off" ) ;
}
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" ) ;
}
}
```

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.

```
# include <stdio.h>
# define _BV(x) ( 1 << x )
void showbits ( unsigned char n );
int main( )
{
unsigned char color, i ;
int c ;
char *rbcolors[ ] = { "Violet", "Indigo", "Blue", "Green",
"Yellow", "Orange", "Red" } ;
printf ( "\nEnter any number: " ) ;
scanf ( "%d", &c ) ;
color = ( unsigned char ) c ;
```

```
printf ( "Colors represented are:\n" ) ;
for ( i = 0 ; i <= 6 ; i++ )
{
if ( ( color & _BV ( i ) ) == _BV ( i ) )
printf ( "%s\n", rbcolors[ i ] ) ;
}
return 0 ;
}
```

Output:

```
Enter any number: 3
Colors represented are:
Violet
Indigo
```

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.


```
# 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

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.

```
#include <stdio.h>

int countSetBits(int n) {
    int count = 0;
    while (n) {
        count += n & 1;
        n >>= 1;
    }
    return count;
}

int main() {
    int game;
    const char *games[] = {
        "Cricket", "Basketball", "Football", "Hockey", "Lawn Tennis",
        "Table Tennis", "Carom", "Chess"
```

```
    };
    printf("Enter the number representing the games won (bit pattern): ");
    scanf("%d", &game);
    int count = countSetBits(game);
    if (count >= 5) {
        printf("The college won the Champion of Champions trophy!\n");
    } else {
        printf("The college did not win the Champion of Champions trophy.\n");
    }
    printf("Games won by the college:\n");
    for (int i = 0; i < 8; i++) {
        if (game & (1 << i)) {
            printf("%s\n", games[i]);
        }
    }
    return 0;
}
```

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 } ;

```
#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;
}
```

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.

```
#include <stdio.h>
int main() {
    int data[] = { 273, 548, 786, 1096 };
    const int YEAR_MASK = 0b1111;
    const int STREAM_MASK = 0b11110000;
    const char* years[] = { "Ist", "IInd", "IIIrd", "IVth" };
    const char* streams[] = { "Mechanical", "Chemical", "Electronics", "CS" };
    for (int i = 0; i < 4; ++i) {
        int year = data[i] & YEAR_MASK;
        int stream = (data[i] & STREAM_MASK) >> 4;
        int room = data[i] >> 8;
        printf("Student %d:\n", i + 1);
        printf("Year: %s\n", years[year]);
        printf("Stream: %s\n", streams[stream]);
        printf("Room Number: %d\n", room);
        printf("\n");
    }
```

```
    return 0;
}
```

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 ;
}
```

k = 51 l = -52 m = 0

n = 97 o = 260 p = 0

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

01011010 11000011
1010101001110101 1111000001011010

```
#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;
}
```

138: Rewrite the following expressions using bitwise compound assignment operators:

a = a | 3 a = a & 0x48 b = b ^ 0x22 c = c << 2

```
a |= 3;
b &= 0x48;
c ^= 0x22;
c <<= 2;
```

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.

```
#include <stdbool.h>
bool checkbits(unsigned int x, int p, int n) {
    unsigned int mask = (1 << n) - 1 << (p - n + 1);
    return (x & mask) == mask;
}
```

```
int main() {
    unsigned int x = 0b11111110;
    int p = 4;
    int n = 3;
    bool result = checkbits(x, p, n);
    if (result) {
        printf("Bits %d to %d are all turned on in number x.\n", p, p - n + 1);
    } else {
        printf("Not all bits %d to %d are turned on in number x.\n", p, p - n + 1);
    }
    return 0;
}
```

140: Write a program to scan an 8-bit number into a variable and check whether its 3rd, 6th and 7th bit is on.

```
#include <stdio.h>
#include <stdbool.h>
int main() {
    unsigned char num;
    bool bit3, bit6, bit7;
    printf("Enter an 8-bit number: ");
    scanf("%hhu", &num);
    bit3 = (num & (1 << 2)) != 0;
    bit6 = (num & (1 << 5)) != 0;
    bit7 = (num & (1 << 6)) != 0;
    if (bit3 && bit6 && bit7) {
        printf("The 3rd, 6th, and 7th bits are all on.\n");
    } else {
```

```
        printf("The 3rd, 6th, and 7th bits are not all on.\n");
    }
    return 0;
}
```

141: 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;
    printf("Enter an unsigned 16-bit integer: ");
    scanf("%hu", &num);
    unsigned short int exchanged_num = ((num & 0xFF) << 8) | ((num >> 8) & 0xFF);
    printf("Original number: %hu\n", num);
    printf("Exchanged number: %hu\n", exchanged_num);
    return 0;
}
```

142) 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;
    printf("Enter an 8-bit number: ");
    scanf("%hhu", &num);
    unsigned char exchanged_num = ((num & 0x0F) << 4) | ((num & 0xF0) >> 4);
    printf("Original number: %hhu\n", num);
    printf("Exchanged number: %hhu\n", exchanged_num);
```

```
    return 0;
}
```

143) 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);
    unsigned char modified_num = num | 0xAA; // 0xAA in binary: 10101010
    printf("Original number: %hhu\n", num);
    printf("Modified number: %hhu\n", modified_num);
    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>
int main() {
    unsigned char num;
    printf("Enter an 8-bit number: ");
    scanf("%hhu", &num);
    if ((num & (1 << 2)) && (num & (1 << 4))) {
        num &= ~(1 << 2);
        num &= ~(1 << 4);
        printf("3rd and 5th bits were on. They are turned off now.\n");
    } else {
```

```
        printf("3rd and 5th bits were not both on.\n");
    }
    printf("Original number: %hhu\n", num);
    return 0;
}
```

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.

```
#include <stdio.h>
int main() {
    unsigned char num;
    printf("Enter an 8-bit number: ");
    scanf("%hhu", &num);
    if (!(num & (1 << 2)) && !(num & (1 << 4))) {
        num |= (1 << 2);
        num |= (1 << 4);
        printf("3rd and 5th bits were off. They are turned on now.\n");
    } else {
        printf("3rd and 5th bits were not both off.\n");
    }
    printf("Original number: %hhu\n", num);
    return 0;
}
```

146) Rewrite the showbits( ) function used in this chapter using the _BV macro.

```
#include <stdio.h>
#define _BV(bit) (1 << (bit))
void showbits(unsigned char num) {
```

```
    int bit;
    for (bit = 7; bit >= 0; bit--) {
        if (num & _BV(bit))
            printf("1");
        else
            printf("0");
    }
    printf("\n");
}
int main() {
    unsigned char num;
    printf("Enter an 8-bit number: ");
    scanf("%hhu", &num);
    printf("Binary representation: ");
    showbits(num);
    return 0;
}
```

147) Write a code to force the compiler to explicitly convert the value of an expression to a particular data type.

```
# include <stdio.h>
int main( )
{
float a, b ;
int x = 6, y = 4 ;
a = x / y ;
printf ( "Value of a = %f\n", a ) ;
b = ( float ) x / y ;
printf ( "Value of b = %f\n", b ) ;
return 0 ;
}
```

Output:

```
Value of a = 1.000000
Value of b = 1.500000
```

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.

```
# include <stdio.h>
float fun1 ( int, int ) ;
float fun2 ( int, int ) ;
float fun3 ( int, int ) ;
float fun1 ( int i, int j )
{
printf ( "In fun1\n" ) ; return 1.0f ;
}
float fun2 ( int i, int j )
{
printf ( "In fun2\n" ) ; return 2.0f ;
}
float fun3 ( int i, int j )
{
printf ( "In fun3\n" ) ; return 3.0f ;
}
int main( )
{
float ( *ptr[ 3 ] ) ( int, int ) ;
float f ; int i ;
ptr[ 0 ] = fun1 ; ptr[ 1 ] = fun2 ; ptr[ 2 ] = fun3 ;
for ( i = 0 ; i < 3 ; i++ )
{
f = ( *ptr[ i ] )( 100, i ) ;
printf ( "%f\n", f ) ;
}
return 0 ;
}
```

Output:

```
In fun1
1.000000
In fun2
2.000000
In fun3
3.000000
```

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.

```
# include <stdio.h>
# include <stdarg.h>
int findavg ( int, ... ) ;
int main( )
{
int avg ;
avg = findavg ( 5, 23, 15, 1, 92, 50 ) ;
printf ( "avg = %d\n", avg ) ;
avg = findavg ( 3, 100, 30, 29 ) ;
printf ( "avg = %d\n", avg ) ;
return 0 ;
}
int findavg ( int tot_num, ... )
{
int avg, i, num, sum ;
va_list ptr ;
va_start ( ptr, tot_num ) ;
sum = 0 ;
for ( i = 1 ; i <= tot_num ; i++ )
{
num = va_arg ( ptr, int ) ;
sum = sum + num ;
}
return ( sum / tot_num ) ;
}
```

Output:

```
avg = 36
avg = 53
```

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 ;
}
```

(a) 0 1 2

(b) 1.000000

4