2203A51540

BAISA SWARNA

(J1)

1(a)

```
#include<stdio.h>
struct employee
{
    int id,salary;
    char name[25];
}emp[100];

void main()
{
    int i,n;
    printf("Enter the no of employees :");
    scanf("%d",&n);
    printf("Enter employee id , name , salary :\n");
    for(i=0;i<n;i++)
    {
        scanf("%d %s %d",&emp[i].id,emp[i].name,&emp[i].salary);
    }

printf("\nEMP_ID\t\tEMP_NAME\t\tEMP_SAL\n********************************
******************\n");
    for(i=0;i<n;i++)
    {
        printf("%d\t\t%s\t\t%d\n",emp[i].id,emp[i].name,emp[i].salary);
    }
}
```

1(b)

```
#include <stdio.h>
struct employee {
    char name[50];
    int id;
    float salary;
} emp[10];

int main() {
    int i, n;
    float total_salary = 0;

    printf("Enter the number of employees: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("\nEnter details of employee %d:\n", i + 1);
        scanf("%s %d %d", emp[i].name,&emp[i].id,&emp[i].salary);
```

```
        total_salary+=emp[i].salary;
    }

    printf("\nTotal salary of all employees = %.2f\n", total_salary);

    return 0;
}
```

2(a)

```
/*Define a structure named "Address" with members:street(string),city
(string),and
pin(integer). Declare another structure named "Person" with members: name
(string) and
address (of type Address). Create a variable of type Person and
initialize its members.
Print the person's name and address details.*/
#include<stdio.h>

//1. define address structure and person structure.
struct person
{
      char name[15];

      struct address
      {
            char street[11];
            char city[20];
            int pin;
      }a; //nested structure

};
void main()
{
      //2.create a variable for person structure
      struct person p;

      //4.1 inserting the values into members of structure dynamically (.
dot operator)
      printf("\nEnter Person Name : ");
      gets(p.name);
      printf("Enter Street : ");
      gets(p.a.street);
      printf("Enter City : ");
      gets(p.a.city);
      printf("Enter Pin: ");
      scanf("%ld",&p.a.pin);
      //4.2 retrieving and printing the values from members of structure
(. dot operator)
      printf("\nEmployee information\n--------------------");
      printf("\nname : %s",p.name);
      printf("\nphone : %s",p.a.street);
      printf("\ncity : %s",p.a.city);
      printf("\npin : %ld",p.a.pin);
}
```

2(b)

```
/*Define a structure named "Address" with members:street(string),city
(string),and
pin(integer). Declare another structure named "Person" with members: name
(string) and
address (of type Address). Create a variable of type Person and
initialize its members.
Print the person's name and address details.Extend the above program to
input details of
multiple persons and their addresses using an array of structures. Print
the details of all
 persons.*/
#include<stdio.h>
// define person and address structure
struct person
{
      char name[15];

      struct address
      {
            char street[11];
            char city[20];
            int pin;
      }a; //nested structure

}p[100];
int main()
{
      int i,n;
      printf("Enter the no. of persons :");
    scanf("%d",&n);
      printf("Enter person name,street,city,pin:\n");
    for(i=0;i<n;i++)
    {
        scanf("%s %s %s
%d",p[i].name,p[i].a.street,p[i].a.city,&p[i].a.pin);
    }

printf("\nP_NAME\t\tP_STREET\t\tP_CITY\t\tP_PIN\n**********************
****************************************\n");
    for(i=0;i<n;i++)
    {

printf("%s\t\t%s\t\t%s\t\t%d\n",p[i].name,p[i].a.street,p[i].a.city,p[i].
a.pin);
    }
}
```

3(a)

```
/*Define a structure named "Book" with members: title (string), author
(string),
price (double) and rating (float). Declare an array of structures to
store information
about multiple books. Allow the user to input the details of the books
and display them.*/
```

```
#include <stdio.h>
struct Book {
    char title[50];
    char author[50];
    double price;
    float rating;
};
int main() {
    struct Book book[100];
    int i, n;
    printf("Enter the number of books: ");
    scanf("%d", &n);
    for(i=0;i<n;++i)
      {
        printf("Enter book title,author,price,rating:\n ");
        scanf("%s %s %lf %f",
book[i].title,book[i].author,&book[i].price,&book[i].rating);
    }
    printf("\nB O O
K\nTITLE\t\tAUTHOR\t\tPRICE\t\tRATING\n**********************************
****************************************\n");
    for(i=0;i<n;++i)
      {

      printf("%s\t\t%s\t\t%lf\t\t%f",book[i].title,book[i].author,book[i]
.price,book[i].rating);
    }
    return 0;
}
```

3(b)

```
/*Write a C program to find the book with the highest rating and lowest
price from
the array of structures.*/
#include <stdio.h>
#include <stdlib.h>

struct book {
    char title[50];
    char author[50];
    float price;
    int rating;
};

int main() {
    struct book b[5] = {
        {"The Alchemist", "Paulo Coelho", 10.0, 4},
        {"The Da Vinci Code", "Dan Brown", 12.0, 3},
        {"The Catcher in the Rye", "J.D. Salinger", 8.0, 5},
        {"To Kill a Mockingbird", "Harper Lee", 9.0, 4},
        {"The Great Gatsby", "F. Scott Fitzgerald", 11.0, 3}
    };

    struct book *highest_rating = &b[0];
    struct book *lowest_price = &b[0];
    int i;
```

```
    for (i = 1; i < 5; i++) {
        if (b[i].rating > highest_rating->rating) {
            highest_rating = &b[i];
        }

        if (b[i].price < lowest_price->price) {
            lowest_price = &b[i];
        }
    }

    printf("Book with highest rating: %s\n", highest_rating->title);
    printf("Book with lowest price: %s\n", lowest_price->title);

    return 0;
}
```

---

(4)
In C programming, a pointer to a structure and a pointer to an array are different types of pointers that point to different data structures.

Pointer to a structure:
****
1)A pointer to a structure holds the memory address of a single structure variable.
2)It allows accessing the members of the structure using the arrow operator (->) to dereference the pointer and access the members.
Example:

```
struct Book {
    char title[50];
    float price;
    int rating;
};
struct Book book1;
struct Book *ptr;
ptr = &book1;
printf("Title: %s\n", ptr->title);
printf("Price: %.2f\n", ptr->price);
printf("Rating: %d\n", ptr->rating);
```

Pointer to an array:
****
1)A pointer to an array holds the memory address of the first element of the array.
2)It allows accessing array elements using pointer arithmetic, where the pointer is incremented or decremented to navigate through the elements.
Example:

```
int numbers[5] = {1, 2, 3, 4, 5};
int *ptr;
ptr = numbers;
printf("%d\n", *ptr);
ptr++;
printf("%d\n", *ptr);
```

5(a)

```
/*Define a structure named "Student" with members: name (string), roll
(integer), and marks
 (float). Create a pointer to the structure and dynamically allocate
memory for it. Input
 the details of the student using the pointer and display them.*/
#include <stdio.h>
#include <stdlib.h>

struct Student {
    char name[50];
    int roll;
    float marks;
};

int main() {
    struct Student *s;
    s = (struct Student*) malloc(sizeof(struct Student));
    printf("Enter Student Name,Roll number,Marks: \n");
    scanf("%s",s->name,&s->roll,&s->marks);
    printf("Displaying Information:\n-----------------------------------
\n");
    printf("Name: %s\n",s->name);
    printf("Roll number: %d\n",s->roll);
    printf("Marks: %.2f\n",s->marks);

    free(s);
    return 0;
}
```

5(b)

```
/*Write a C program to sort an array of structures using pointers. Sort
the array based on
 roll numbers in ascending order and display the sorted list.*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct student {
    int roll;
    char name[50];
} s[5];
void sort(struct student *s, int n)
{
    int i, j;
    struct student temp;
    for(i=0;i<n;i++)
      {
        for (j=i+1;j<n;j++)
            {
             if((s+i)->roll>(s+j)->roll)
                  {
                temp=*(s+i);
```

```
                *(s+i)=*(s+j);
                *(s+j)=temp;
            }
        }
    }
}
int main()
{
    int i,n=5;
    struct student *p;
    p=s;

    for(i=0;i<n;i++)
      {
        printf("Enter Roll number of Student %d: ",i+1);
        scanf("%d",&(p+i)->roll);
        printf("Enter Name of Student %d: ",i+1);
        scanf("%s",(p+i)->name);
    }
    sort(p,n);
    printf("\nSorted list:\n");
    for(i=0;i<n;i++)
      {
        printf("%d\t%s\n",(p+i)->roll,(p+i)->name);
    }
    return 0;
}
```

---

(6)
In C programming, the two main types of files are text files and binary files. The key difference between them lies in how the data is stored and interpreted.

Text Files :
****
1)Text files store data as a sequence of characters. Each character is represented using an encoding scheme, such as ASCII or UTF-8.
2)The contents of text files are human-readable and can be opened and edited using text editors.
3)Text files typically contain plain text, such as letters, numbers, symbols, and newline characters.
4)They are read and written using text-oriented I/O functions like fopen, fprintf, fscanf, fgets, and fputs.
5)Examples: source code files (.c, .h), configuration files, log files, and text-based data files (CSV, XML, JSON).

Binary Files :
****
1)Binary files store data in its raw binary format. The data is not interpreted as characters but as a series of bytes.
2)The contents of binary files are not intended to be human-readable and may contain any type of data, including binary data, structures, or serialized objects.
3)They can store complex data structures without any concern for character encoding or formatting.

4)They files are read and written using binary-oriented I/O functions like fopen, fwrite, fread, fseek, and ftell.
5)Examples: image files (BMP, JPEG), audio files (WAV, MP3), video files (MP4, AVI), and compiled object files (.o, .dll).

7(a)

```
/*Write a C program to create a text file named "data.txt" and allow the
user to enter text.
Append the entered text to the file.*/
#include <stdio.h>
#include <stdlib.h>
int main() {
    FILE *fp;
    char text[100];
    // Open the file in append mode
    fp = fopen("data.txt", "a");
    if (fp == NULL)
      {
        printf("Error opening file!\n");
        exit(1);
    }
    // Ask the user to enter text
    printf("Enter text: ");
    fgets(text, 100, stdin);
    // Append the text to the file
    fprintf(fp, "%s", text);
    // Close the file
    fclose(fp);
    return 0;
}
```

7(b)

```
/*Write a C program to read the contents of a text file named "data.txt"
and display them on the
 console*/
#include <stdio.h>
int main() {
    FILE *fp;
    char ch;
    // Open the file in read mode
    fp = fopen("data.txt", "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
    // Read the file character by character and display it
    while ((ch=fgetc(fp))!= EOF)
      {
        printf("%c", ch);
    }
    // Close the file
    fclose(fp);
    return 0;
}
```

7(c)

```
/*Write a C program to create a binary file named "students.dat" to store
information about
students (name, roll, and marks). Allow the user to input the details of
multiple students and
write them to the file.*/
#include <stdio.h>
#include <stdlib.h>

struct student {
    char name[50];
    int roll;
    float marks;
} s;
int main() {
    FILE *fptr;

    fptr = fopen("students.dat", "wb");
    if (fptr == NULL) {
        printf("Error opening file");
        exit(1);
    }
    int n,i;
    printf("Enter number of students: ");
    scanf("%d", &n);
    for(i=0;i<n;i++)
      {
        printf("For Student %d\nEnter name: ",i+1);
        scanf("%s", s.name);
        printf("Enter roll number: ");
        scanf("%d", &s.roll);
        printf("Enter marks: ");
        scanf("%f", &s.marks);
        fwrite(&s, sizeof(s), 1, fptr);
    }
    fclose(fptr);
    return 0;
}
```

7(d)

```
/*Write a C program to randomly access and read the details of a student
from the binary file
"students.dat" based on the roll number entered by the user.*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct student {
    int roll;
    char name[50];
    int age;
};
int main()
```

```
{
    FILE *fp;
    struct student s;
    int n,rollnumber,found=0;
    fp = fopen("students.dat", "rb");
    if (fp == NULL) {
        printf("Error opening file\n");
        exit(1);
    }
    printf("Enter roll number of student to search: ");
    scanf("%d", &rollnumber);
    while (fread(&s, sizeof(struct student), 1, fp))
      {
        if (s.roll == rollnumber)
            {
            printf("Roll Number: %d\nName: %s\nAge:
%d\n",s.roll,s.name,s.age);
            found = 1;
            break;
        }
    }
    if (!found)
      {
        printf("Record not found\n");
    }
    fclose(fp);
    return 0;
}
```

---

(8)

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

struct file {
    char name[50];
    int size;
    char content[100];
};

void createFile() {
    struct file f;
    FILE *fp;

    printf("Enter file name: ");
    scanf("%s", f.name);

    printf("Enter file size: ");
    scanf("%d", &f.size);

    printf("Enter file content: ");
    scanf("%s", f.content);

    fp = fopen(f.name, "w");
    fwrite(&f, sizeof(f), 1, fp);
    fclose(fp);
}
```

```
void readFile() {
    struct file f;
    FILE *fp;

    printf("Enter file name: ");
    scanf("%s", f.name);

    fp = fopen(f.name, "r");
    fread(&f, sizeof(f), 1, fp);
    printf("Name: %s\nSize: %d\nContent: %s\n", f.name, f.size,
f.content);
    fclose(fp);
}

void updateFile() {
    struct file f;
    FILE *fp;

    printf("Enter file name: ");
    scanf("%s", f.name);

    printf("Enter new file content: ");
    scanf("%s", f.content);

    fp = fopen(f.name, "w");
    fwrite(&f, sizeof(f), 1, fp);
    fclose(fp);
}

void deleteFile() {
    char name[50];

    printf("Enter file name: ");
    scanf("%s", name);

    if (remove(name) == 0) {
        printf("File deleted successfully.\n");
    } else {
        printf("Unable to delete the file.\n");
    }
}

int main() {
    int choice;

    while (1)
      {
        printf("\nFile Management System:\n");
        printf("1. Create file\n");
        printf("2. Read file\n");
        printf("3. Update file\n");
        printf("4. Delete file\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
```

```
            {
             case 1:
                 createFile();
                 break;
             case 2:
                 readFile();
                 break;
             case 3:
                 updateFile();
                 break;
             case 4:
                  deleteFile();
                  break;
      }
    }
}
```

9(a)

```
/*Define an enumeration named "Month" with constants representing the
months of the year.
 Write a C program to input a month number from the user and display the
corresponding
 month name using the enumeration.*/
#include <stdio.h>

enum Month {
    JANUARY = 1,
    FEBRUARY,
    MARCH,
    APRIL,
    MAY,
    JUNE,
    JULY,
    AUGUST,
    SEPTEMBER,
    OCTOBER,
    NOVEMBER,
    DECEMBER
};

int main()
{
    int monthNumber;

    printf("Enter the month number (1-12): ");
    scanf("%d", &monthNumber);

    if (monthNumber>=JANUARY && monthNumber<=DECEMBER)
      {
        enum Month month=monthNumber;

        switch(month)
            {
             case JANUARY:
                 printf("January\n");
                 break;
```

```
            case FEBRUARY:
                printf("February\n");
                break;
            case MARCH:
                printf("March\n");
                break;
            case APRIL:
                printf("April\n");
                break;
            case MAY:
                printf("May\n");
                break;
            case JUNE:
                printf("June\n");
                break;
            case JULY:
                printf("July\n");
                break;
            case AUGUST:
                printf("August\n");
                break;
            case SEPTEMBER:
                printf("September\n");
                break;
            case OCTOBER:
                printf("October\n");
                break;
            case NOVEMBER:
                printf("November\n");
                break;
            case DECEMBER:
                printf("December\n");
                break;
        }
    }
      else
      {
        printf("Invalid month number.\n");
    }
    return 0;
}
```

9(b)

```
#include <stdio.h>

enum Month
{
   JANUARY = 1,
    FEBRUARY,
    MARCH,
    APRIL,
    MAY,
    JUNE,
    JULY,
    AUGUST,
    SEPTEMBER,
```

```
    OCTOBER,
    NOVEMBER,
    DECEMBER
};

int main() {
   int month;
   printf("Enter month number (1-12): ");
   scanf("%d", &month);

   switch(month) {
      case JANUARY:
         printf("January has 31 days.\n");
         break;
      case FEBRUARY:
         printf("February has 28 or 29 days.\n");
         break;
      case MARCH:
         printf("March has 31 days.\n");
         break;
      case APRIL:
         printf("April has 30 days.\n");
         break;
      case MAY:
         printf("May has 31 days.\n");
         break;
      case JUNE:
         printf("June has 30 days.\n");
         break;
      case JULY:
         printf("July has 31 days.\n");
         break;
      case AUGUST:
         printf("August has 31 days.\n");
         break;
      case SEPTEMBER:
         printf("September has 30 days.\n");
         break;
      case OCTOBER:
         printf("October has 31 days.\n");
         break;
      case NOVEMBER:
         printf("November has 30 days.\n");
         break;
      case DECEMBER:
         printf("December has 31 days.\n");
         break;
      default:
         printf("Invalid month number. Please try again.\n");
   }
   return 0;
}
```

---

(10)

```
/*Write a C program that takes two command line arguments: an integer and
a filename. The
```

```
 program should read the contents of the given file and print the first
'n' lines, where
 'n' is the integer provided as a command line argument.*/
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    FILE *fp;
    char *line = NULL;
    size_t len = 0;
    ssize_t read;
    int i,n = atoi(argv[1]);
    fp = fopen(argv[2], "r");
    if (fp == NULL)
      {
        printf("Could not open file %s", argv[2]);
        return 1;
    }
    for (i=0;i<n;i++)
      {
        if((read=getline(&line,&len,fp))!=-1)
            {
            printf("%s", line);
        }
        else {
            break;
        }
    }
    fclose(fp);
    if (line) {
        free(line);
    }
    return 0;
}
```