1)Nested structure for student details:

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

#define MAX_DEPARTMENTS 10
#define MAX_STUDENTS 100

typedef struct Student_Details
{
   char name[50];
   char mailID[50];
   char mobileNumber[50];
   char percentage[50];
}Student_Details;

typedef struct Department
{
   char deptName[50];
   struct Student_Details students[MAX_STUDENTS];
   int numStudents;
   float totalAverage;
}Department;

int main()
{
   int ips;
   printf("Number of Inputs: ");
   scanf("%d", &ips);

   Department departments[MAX_DEPARTMENTS];
   int depts = 0;

   for (int i = 0; i < ips; i++)
   {
      char name[50], mailID[50], mobileNumber[50], percentage[50],
deptName[50];
      printf("Enter student details (Name, mailID, mobile number, percentage,
department):\n");
      scanf("%s %s %s %s %s", name, mailID, mobileNumber, percentage,
deptName);

      int index = -1;
      for (int j = 0; j < depts; j++)
```

```
        {
          if (strcmp(departments[j].deptName, deptName) == 0)
            {
              index = j;
              break;
            }
        }
        if (index == -1)
        {
            index = depts;
            strcpy(departments[depts].deptName, deptName);
            departments[depts].numStudents = 0;
            departments[depts].totalAverage = 0;
            depts++;
        }

        Student_Details student;
        strcpy(student.name, name);
        strcpy(student.mailID, mailID);
        strcpy(student.mobileNumber, mobileNumber);
        strcpy(student.percentage, percentage);
        departments[index].students[departments[index].numStudents] =
student;
        departments[index].numStudents++;

        float totalPercentage = 0;
        for (int j = 0; j < departments[index].numStudents; j++)
          {
            char *percentage = departments[index].students[j].percentage;
            int percentageValue = atoi(percentage);
            totalPercentage += percentageValue;
        }
        departments[index].totalAverage = totalPercentage /
departments[index].numStudents;
    }

    printf("Number of branches: %d\n", depts);
    printf("Average percentage per Department:\n");

    for (int i = 0; i < depts; i++)
    {
        printf("%s - %.0f%%\n", departments[i].deptName,
departments[i].totalAverage);
    }
```

```
}
```
2)Structure to display date and time:

```
#include <stdio.h>

typedef struct Time {
    short hours;
    short minutes;
    short seconds;
    short day;
    short month;
    short year;
}Time;

int main() {
   Time t;
   printf("Enter the values \n");
   scanf("%hu %hu %hu %hu %hu %hu", &t.hours, &t.minutes, &t.seconds,
&t.day, &t.month, &t.year);
   printf("Time : %hu:%hu:%hu\n", t.hours, t.minutes, t.seconds);
   printf("Date : %hu-%hu-%hu\n", t.day, t.month, t.year);
   printf("Size of struct: %lu bytes\n", sizeof(t));
   return 0;
}
```

3) Write a program to get the input from the file and create a new encrypted file and then read the encrypted file and decrypt the content.

Sol:

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

#define MAX_SIZE 1000

void encrypt(char *input_file, char *output_file);
void decrypt(char *input_file, char *output_file);

main()
{
   char choice, input_file[MAX_SIZE], output_file[MAX_SIZE];
   do
   {
      printf("Enter E to encrypt, D to decrypt, or Q to quit: ");
      scanf(" %c", &choice);
      switch (choice)
```

```
        {
            case 'E':
            case 'e':
                printf("Enter the file to be encrypted: ");
                scanf("%s", input_file);
                printf("Enter the name of the encrypted file: ");
                scanf("%s", output_file);
                encrypt(input_file, output_file);
                printf("Encrypted file is %s\n", output_file);
                break;
            case 'D':
            case 'd':
                printf("Enter the file to be decrypted: ");
                scanf("%s", input_file);
                printf("Enter the name of the decrypted file: ");
                scanf("%s", output_file);
                decrypt(input_file, output_file);
                printf("Decrypted file is %s\n", output_file);
                break;
            case 'Q':
            case 'q':
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
                break;
        }
    } while (choice != 'Q' && choice != 'q');
}

void encrypt(char *input_file, char *output_file)
{
    FILE *input, *output;
    char ch;
    input = fopen(input_file, "r");
    output = fopen(output_file, "w");
    if (input == NULL || output == NULL)
    {
        printf("Error opening file!\n");
        exit(1);
    }
    while ((ch = fgetc(input)) != EOF)
    {
        ch += 7;
```

```
        fputc(ch, output);
    }
    fclose(input);
    fclose(output);
}

void decrypt(char *input_file, char *output_file)
{
    FILE *input, *output;
    char ch;
    input = fopen(input_file, "r");
    output = fopen(output_file, "w");
    if (input == NULL || output == NULL)
    {
        printf("Error opening file!\n");
        exit(1);
    }
    while ((ch = fgetc(input)) != EOF)
    {
        ch -= 7;
        fputc(ch, output);
    }
    fclose(input);
    fclose(output);
}
```