1. Why do we need data files? What are the different file opening modes? Write a program that reads data from a file "input.txt" and writes to "output.txt" file.

   Ans:

   1st Part:

   The basic I/O operations like printf() and scanf() are console oriented functions, which always requires the use of keyboard for input and a computer screen for output. These functions read, writes and stores data to temporary storage of computer which gets wiped out as soon as the program is terminated, or the computer is turned off. These functions are worthwhile for very small-scale operation where data doesn't need to be permanently stored. But it becomes cumbersome and time consuming to handle a large volume of data. Also, at some point these data needs to be accessed and modified, which can't be done as they are temporary. These problems gave birth to concept of data files.

   The data file allows us to store information permanently and to access and alter that information whenever necessary. Therefore, if we need to store the data for long term and in a format which allows us to access and alter those data, we need data files.

   2nd Part:

   The different file opening modes are as follows:

i. "r" (read)

   This opens an existing file for reading purpose only. It looks for specified file. If the file exists, it loads it into a memory and sets up a pointer to the file. If the file isn't found, it returns NULL.

ii. "w" (write)

   This mode opens a file for writing purpose. It searches for the specified file. If the file exists, the content of the file is overwritten. If the file doesn't exist, a new fie is created implicitly. If it fails to open the fie, it returns NULL.

iii. "a" (append)

   This mode opens an existing file to appending purpose i.e., the new information is added after the existing information. It searches for the specified file. If the file exists, it loads it into a memory and sets up a pointer to the last character of file. If the file doesn't exist, a new fie is created implicitly. If it fails to open the fie, it returns NULL.

iv. "r+" (read + write)

   This opens an existing file for reading and writing purpose. It looks for specified file. If the file exists, it loads it into a memory and sets up a pointer to the file. If the file isn't found, it returns NULL, i.e., the file must already exist. After reading the content of the file, new content can be written to the file. The new content will overwrite the previous content.

v. "w+" (write + read)

   This opens a file for writing and reading purpose. It looks for specified file. If the file exists, it loads it into a memory and sets up a pointer to the file. The previous content is destroyed. If the file isn't found, a new file is created implicitly. If the operation fails, it returns NULL. After writing the content to the file, new content can be read from the file.

vi. "a+" (append + read)

   This opens an existing file for appending and reading purpose. It looks for specified file. If the file exists, it loads it into a memory and sets up a pointer to the last character of file. Then the new content can be added to the already existing content. If the file isn't found, a new file is created implicitly. If the operation fails, it returns NULL. After appending the new content to the file, those can be read from the file.

3rd Part:

Program:

```c
#include <stdio.h>
int main(){
    FILE *fptrSource, *fptrDest;
    char ch;
    fptrSource = fopen("files\\input.txt", "r");
    fptrDest = fopen("files\\output.txt", "w");

    if (fptrSource == NULL){
        printf("\ninput.txt can not be opened.");
        exit(1);
    }
    printf("\ninput.txt opened successfully.");

    if (fptrDest == NULL){
        printf("\noutput.txt can not be created.");
        exit(1);
    }
    printf("\noutput.txt created successfully.");

    while((ch = fgetc(fptrSource))!= EOF){
        fputc(ch, fptrDest);
    }
    printf("\nSuccessfully copied from 'input.txt' to 'ouput.txt'.");

    fclose(fptrSource);
    fclose(fptrDest);
    return 0;
}
```

2. Explain different file I/O functions with example.

Ans:

The different I/O functions of file are as follows:

i. fgets() : It is used to read string from file.
Syntax: `fgets(string_variable, int_value, file_ptr_variable);`

ii. fputs() : It is used to write a string to a file.
Syntax: `fputs(string_variable, file_ptr_variable);`

iii. fgetc() : It is used to read a character from a file.
Syntax: `fgetc(file_ptr_variable);`

iv. fputc() : It is used to write a character to a file.
Syntax: `char_variable = fputc(char_variable, file_ptr_variable);`

v. fprintf() : It is a formatted output function which is used to write integer, float, char or string data to a file.
Syntax: `fprintf(file_ptr_variable, "control_string", list_variables);`

vi. fscanf() : It is a formatted input function which is used to read integer, float, char or string to a file.
Syntax: `fscanf(file_ptr_variable, "control_string", &list_variables);`

vii. fread() : It is used for record input.
Syntax:
```
fread(&fptr, size_of_array_or_structure, number_of_structure_or_array,
&list_variables);
```

viii. fwrite() : It is used for record output.
Syntax:
```
fread(&fptr, size_of_array_or_structure, number_of_structure_or_array,
&list_variables);
```

Example:

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

int main()
{
    FILE *fptr;
    char ch, str[100], ch_temp, str_temp[100];
    int year, year_temp;

    fptr = fopen("files\\q2.txt", "w+");
    if (fptr == NULL){
        printf("Operation failed !!!");
        exit(1);
    }
    printf("File created and opened successfully.");
```

```c
    ch = 'C';
    fputc(ch, fptr);
    rewind(fptr);
    ch_temp = fgetc(fptr);
    printf("\n%c", ch_temp);
    fclose(fptr);

    fptr = fopen("files\\q2.txt", "w+");
    strcpy(str, "Happy New Year");
    fputs(str, fptr);
    rewind(fptr);
    fgets(str_temp, 30, fptr);
    printf("\n%s", str_temp);
    fclose(fptr);

    fptr = fopen("files\\q2.txt", "w+");
    year = 2021;
    fprintf(fptr, "%d", year);
    rewind(fptr);
    fscanf(fptr, "%d", &year_temp);
    printf("\n%d", year_temp);
    fclose(fptr);

    return 0;
}
```

5. Explain the use of graphical functions. Write a program to draw a triangle using line() graphics function.

Ans:

Part 1:

There are various graphics functions in C. Some of them with their users are explained below:

i. putpixel(x,y,color)

The use of this function is to put a pixel or in other words a dot at position x, y given in inputted argument. Here the whole screen is imagined as a graph. In other words, the pixel at the top left-hand corner of the screen represents the value (0, 0). Here the color is the integer value associated with colors and when specified the picture element or the dot is placed with the appropriate color associated with that integer value.

ii. getpixel(x,y)

This function when invoked gets the color of the pixel specified. The color got will be the integer value associated with that color and hence the function gets an integer value as return value. So the smallest element on the graphics display screen is a pixel or a dot and the pixels are used in this way to place images in graphics screen in C language.

iii. line(x1,y1,x2,y2)

The use of this function is to draw a line from (x1,y1) to (x2,y2).Here also the coordinates are passed taking the pixel (0,0) at the top left hand corner of the screen as the origin. And also one must note that the line formed is by using number of pixels placed near each other

Part 2:

```c
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    line(150, 150, 200, 200);
    line(150, 150, 100, 200);
    line(100, 200, 200, 200);

    getch();
    closegraph();
    return 0;
}
```

6. Discuss with example any five graphics function used in c-programming.

Ans: There are numerous graphics functions available in c-programming. Some of them are listed below:

i. line()
Syntax: `line(int x1, int y1, int x2, int y2);`
It draws a line from point having coordinate x1, y1 to x2, y2 using current settings for the line drawing and current drawing color.

ii. circle()
Syntax: `circle(int x, int y, int r);`
It draws a circle having center point (x, y) and radius r with current color.

iii. ellipse()
Syntax: `ellipse(int x, int y, int startAngle, int endAngle, int xRadius, int yRadius);`
It draws an ellipse with current color, where point (x,y) is the center point of the ellipse, startAngle is starting angle in degree, endAngle is ending angle in degree, xRadius and yRadius are the radius of ellipse along x-axis and y-axis respectively.

iv. arc()
Syntax: `arc(int x, int y, int startAngle, int endAngle, int radius);`
This function is used to draw a circular arc, where point (x,y) is the center of circle, startAngle and endAngle are starting angle and ending angle of arc measured in degree and radius is the radius of the circle.

v. rectangle()
Synatx: `rectangle(int x1, int y1, int x2, int y2);`
It draws rectangle from two end points of diagonal of the rectangle. Where point (x1,y1) is left top corner point of the rectangle (i.e, start point of principal diagonal) and point (x2,y2) is the right bottom corner point of the rectangle (i.e, end point of principal diagonal).

Example:

```
#include <conio.h>
#include <graphics.h>
int main(){
    int gd = DETECT, gm;
    initgraph(&gd,&gm,(char *)"");
    line(150, 150, 200, 200);
    circle(200, 300, 50);
    ellipse(100, 100, 0, 360, 50, 75);
    arc(400, 200, 90, 180, 75);
    rectangle(150, 250, 200, 300);
    getch();
    closegraph();
    return 0;
}
```