Answers

1) Difference between code and an executable file →

i) Code → This refers to the source code written in a programming language. It is human readable & must be converted into machine-readable instruction for the computer.

ii) Executable File → This is the machine readable file generated after compiling the source code. It can be directly run by the operating system.

Step to convert a C code to executable file →

a) Preprocessing → The preprocessor handles directives like #include & #define, resulting in an intermediate file with the macros exporded.

e.g. → gcc -E prg.C -o prg.i

b) Compilation → The preprocessed code is compiled into an assembly code ~~is converted~~ specific to the target machine. e.g. → gcc -s prg.i -o prg.s

c) Assembly → The assembly code is converted into machine code. e.g. → gcc -s prg.i -o prg.s

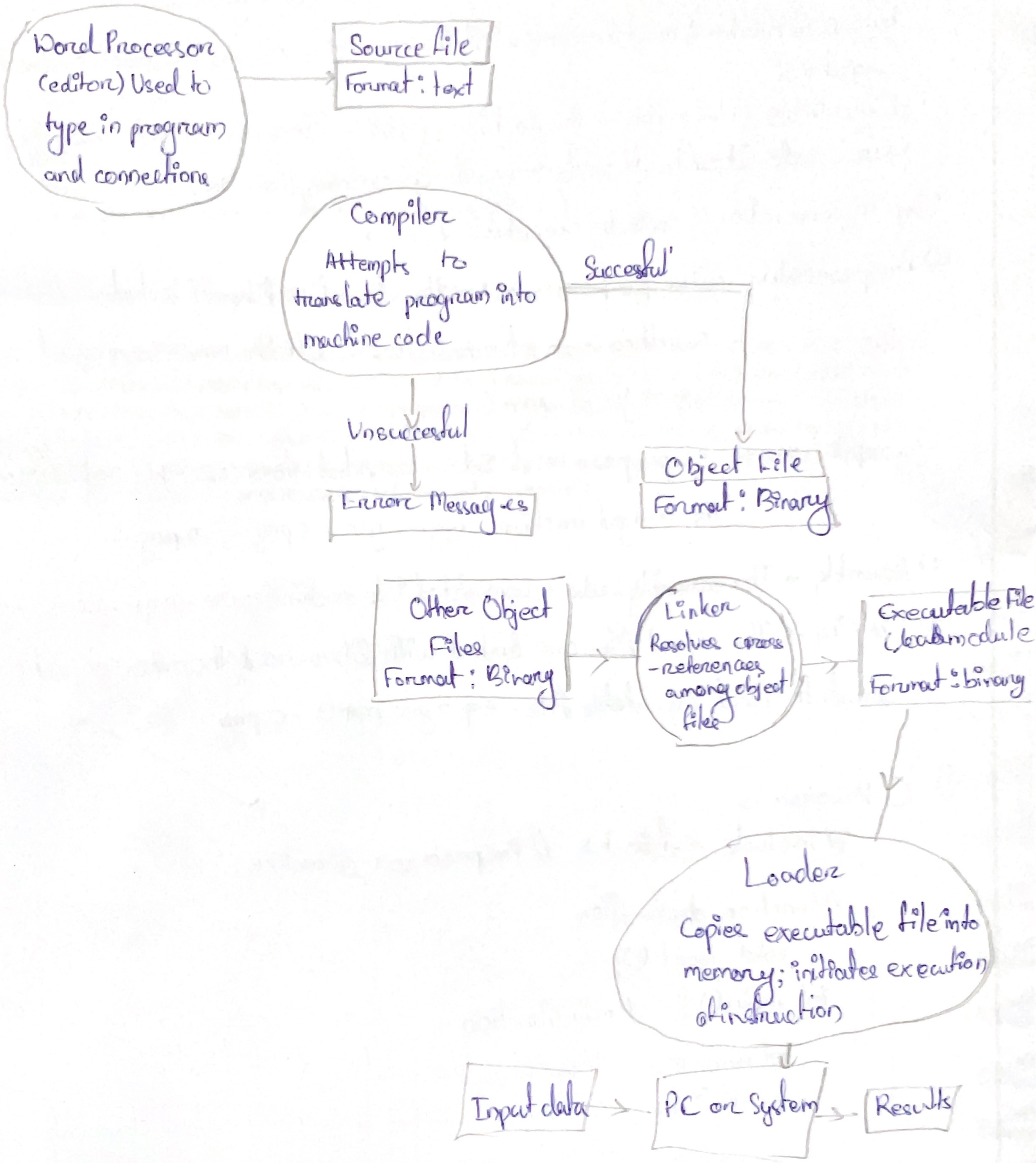
d) Linking → The object files are linked with libraries & the other object files to create the final executable file. e.g → gcc prg.O -o prg

2) C Program →

```
#include <stdio.h> // Preprocessor directive
// Function declaration
    void greet();
int main() {  // main function
    int num = 10;  // Variable declaration & initialisation
    printf("The value of num is : %d \n", num); // Function call
    greet(); // Function call
    return 0; // return statement
}
// Function definition
    void greet() {
        printf("Hello world ! \n");
    }
}
```

Name: —————————— Regd. Number: ——————————

Fig 1.1


Word Processor (editor) Used to type in program and connections
Source file
Format: text
Compiler
Attempts to translate program into machine code
Successful
Unsuccessful
Error Messages
Object File
Format: Binary
Other Object Files
Format: Binary
Linker
Resolves cross-references among object files
Executable File (loadmodule
Format: binary
Loader
Copies executable file into memory; initiates execution of instruction
Input data
PC or System
Results

Parts of C code →

i) Preprocessor directive → #include <stdio.h> : This line is a preprocessor directive that includes the standard input output library stdio.h, which contain functions like printf()

ii) Comments → In C, comments are ignored by the compiler and are used for documenting code.

iii) Function declaration → void greet(); This function declares a function named greet that returns no value (void) and takes no parameters.

iv) Main function →

a) int main() {...} : The main() function is the entry C program execution of a C program begins with the function.

b) int: The return type of main() is int (integer), and it usually returns 0 to indicate successfully execution.

v) Variable declaration & initialisation → int num = 10; This declares an integer variable num and initialise it with the value 10.

vi) Function call →

a) Printf ("The value of num is : %d \n", num); The printf() function is called to print a format string to the console.

b) greet(); This calls the greet() function

vii) Return Statement → Returns (); This statement returns 0 from the main() function, indicating that the program finished successfully.

viii) Function Definition → void greet() {...}. This defines the greet.

3) The maximum value that can be stored in a float in C is approximately 3.4×10^{38}. This value is defined in the <float.h> header as FLT_MAX.

Reason → i) A float in 32 bits, following the IEEE 754 standard.

ii) 1 bit for the sign (positive or negative)

iii) 8 bits for the exponent (range of -127 to 127 after bias)

iv) 23 bits for the fraction

4) The scanf() function returns an integer representing the no. of input items successfully read and assigned.

Possible Return Values →

i) Positive Integer (>0): No. of successfully read input items

e.g → input → 10 20 output → 2

ii) Zero → No matching input for the format

e.g → input abc (expecting a no.)

output → 0

iii) EOF (-1) : End of input on error occurred

e.g. → input → end of file on error

output → -1

5) i) 4294967173 → output

Explanation → %u is used to print an unassigned integer but x is assigned integer (-123)

ii) Output → 1 4

Explanation → size of (ch) returns 1, as chis of type char

size of ('A') returns 4, because 'A' is treated as an int literal in C

iii) Output → 4

Explanation → ch+4 is promoted to int, so size of (ch+4) will return the size of int, typically 4.

iv) Output → 16

Explanation → 4/2 = 2

6*2 = 12

2+2+12 = 16

v) Output → 16

Explanation → 4/2 = 2

6 * 2 = 12

2 + 12 + 2 = 16

vi) Output → Hello
6

Explanation → Printf("Hello\n") prints "Hello" followed by a newline, and it returns the no. of char printed.

vii) Output → 5 == 55.500000 = 55.500000

5 == 5.55000e+02 == 1.234500E+02

11 == 855.5 = 123.45

viii) Output
32 == 32 == 40 == 40

32 == 32 == 040 = 0x20

32 == 32 == 040 = 0x20

+32 == +32 == 032 == 0x45B

ix) Output →

3000.000000	0.003500	10.500000
3.000000e+03	3.500000e-03	1.050000e+01
3.000000E+03	3.500000E-03	1.050000E+01

Explanation → %.f prints floating point no., %.e & %.E print no. in scientific notation

x) Outputs 54321 : : 54321 : : 54321 : : 54321 : 876.543030

876.543030 : : 876.543030 : : 876.543030 : : 876.543030

Explanation → %.3d, %.10d and %.3f & %.10f controls the field width of for printing integers & floats respectively.

6)
```
#include<stdio.h>
int main(){
    float m, kms, cms, mms, feet, in;
    printf("Enter distance in meters: ");
    scanf("%f", &m);
    kms = m*0.001;
    cms = m*100;
    mms = m*1000;
    feet = m*3.28084;
    in = m*39.3701;
    printf("+--------+----------+\n");
    printf("| Unit   | Value    |\n");
    printf("+--------+----------+\n");
    printf("| Meters | %16.2f |\n");
    printf("| Kilometers | %16.3f |\n");
    printf("| Centimeters | %16.2f |\n");
    printf("| Milimeters | %16.2f |\n");
    printf("| Feet | %16.2f |\n");
    printf("| Inches | %16.2f |\n");
    printf("+--------+----------+\n");
}
```

Output → Enter distance in meter : 5

```
+--------+----------+
| Unit   | Value    |
+--------+----------+
| Meters | 5.00     |
| Kilometers | 0.005    |
| Centimeter | 500.00   |
| Milimeter | 5000.00  |
| Feet | 16.40    |
| Inches | 196.85   |
+--------+----------+
```

~~printf("Feet~~

Feet	16.40
Inches	196.85

7) #include <stdio.h>

int main(){

char des_grade;

float min_avg, cur_avg, final_weight, final_score;

printf("Enter the desired grade:");

scanf("%.c", &des_grade);

printf("Enter minimum average required");

scanf("%.f", &min_avg);

printf("Enter current average in courses");

scanf("%.f", &cur_avg);

printf("Enter how much final count as a percentage of the course grade");

scanf("%.f", &final_weight);

final_score = (min_avg - cur_avg * (1- final_weight/100) (final_wei

printf("You need a score of %.2f on the final to get a %.c\n", final_s

des_grade);

return 0;

}

Output →

Enter desired grade > B

Enter minimum average required > 79.5

Enter current average in counts > 74.6

Enter how much final counts as a percentage of the course grade > 25

You need a score of 94.20 on the final to get a B

8) #include <stdio.h>
#include <math.h>
int main(){
float take_off_speed_kmh, take_off_speed_ms, distance, acceleration, time;
printf("%f", &distance);
printf("Enter the jets take off speed (km/h): ";
scanf("%f", &take_off_speed_kmh);
take_off_speed_ms = take_off_speed_kmh * 1000/3600;
acceleration = take_off_speed_ms * take_off_speed_ms)/(2*distance);
time = take_off_speed_ms / acceleration;
printf("The required acceleration is (%.2f) m/s^2 \n", acceleration);
printf("The time required to reach takeoff speed in %.2f seconds \n",
time);
return 0;
}

Output →

Enter the jet's take off speed (km/h): 100
Enter the distance cover which the jet is accelerated (meters): 50
The required acceleration is : 7.72 m/s²
The time required to reach take off speed in 3.60s.

S. Pattnaik
01.10.24