(1)

```
#include<iostream>
using namespace std;

class vehicle {
   string name, fuel_type, model, speed; // Attributes to store vehicle details

public:
   void set();  // Method to set the vehicle details

   void display_max_speed();  // Method to display the maximum speed of the vehicle
};

inline void vehicle::set()
{
   {
      // vehicle name
      cout << "Enter the name of the vehicle: "; // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> name;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)

      // fuel type of the vehicle
      cout << "Enter the fuel type of " << name << " : "; // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> fuel_type;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)

      // Asking for the model of the vehicle
      cout << "Enter the model of " << name << " : "; // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> model;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)

      // max speed of the vehicle
      cout << "Enter the max speed of " << name << " : "; // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> speed;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
   }
}

inline void vehicle::display_max_speed()
{
   // Display the max speed of the vehicle
   cout << "Max speed of " << name << " is " << speed << "."; // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
}
```

```
int main() {
   vehicle a1;  // Create an object of the vehicle class
   a1.set();  // Call the set method to input vehicle details
   a1.display_max_speed();  // Call the method to display max speed
}
```

==========

(2)

```
#include<iostream>
// (it is the header file for input-output stream functions)
/*
Group_Fruits

Grapes, Watermelon, Kiwi, Mandarin,

Color_fruit, Count_items, Size, Texture, Produced_in,

No_of_Ripening_fruits()
Aging_time_of_fruit()
*/

using namespace std;  // (it allows us to use standard library objects without the std:: prefix)

class fruits {
   string name, color, texture, produced_in;  // (attributes to store fruit properties)
   int count_items, size;  // (integer variables to store item count and size)

public:
   void set();  // (method to input fruit details)
   void display();  // (method to display fruit details)
};

inline void fruits::set() {
   {
      cout << "Enter name: ";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> name;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
      cout << "Enter color of it: ";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> color;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
      cout << "Enter count: ";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> count_items;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
```

```
        cout << "Enter size: ";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
        cin >> size;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
        cout << "Enter texture: ";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
        cin >> texture;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
    }
}

inline void fruits::display() {
    {
        cout << "!! FOOD ITEM LIST  !!endl";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
        cout << "Name: " << name << " ,Color: " << color << " ,Item count: " << count_items << " ,Size: " << size << ",Texture: " << texture << endl;  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
    }
}

int main() {
    int n;  // (variable to store the number of fruit entries)
    cout << "enter your food_limit: ";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
    cin >> n;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
    fruits temp;  // (temporary object to set and display fruit information)
    for (int i = 0; i < n; i++) {
        temp.set();  // (calls the set method to input fruit details)
    }
    for (int i = 0; i < n; i++) {
        temp.display();  // (calls the display method to output fruit details)
    }

    return 0;  // (returns 0 to indicate successful program termination)
}
```

==========

(3)

```
#include<iostream>   // compulsory for all the codes to use input-output streams
/* EXERCISE TO CREATE
CLASS NAME: TEST
member variable: int a
member function: void set(), void Display()
*/

using namespace std;  // allows direct use of standard library objects without std:: prefix
```

```
class test {          // creation of class 'test'
   int a;            // member variable declared
public:              // public access specifier
   // The keyword 'public' specifies that everything defined after it is accessible from outside the class.

   void set();       // member function declaration for setting value of 'a'
   void display();   // member function declaration for displaying value of 'a'
   // 'void' function does not return any value.
};                   // body of the class 'test' is closed

// Inline function definition for set()
inline void test::set() {
   cout << "enter the value of A:";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
   cin >> a;                        // (it is an object belongs to istream " >> " refers to bits from keyboard to memory)
}

// Inline function definition for display()
inline void test::display() {
   cout << "the value of A is:" << a << endl;  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
   // 'endl' inserts a newline and flushes the output buffer.
}

int main() {          // main function: program execution starts here
   test t1;           // creation of object 't1' of class 'test' (memory is allocated here)

   t1.set();          // calling set() function to input value of 'a'
   t1.display();      // calling display() function to output value of 'a'

   return 0;          // returns 0 to indicate successful execution
}
```

==========

(4)

```
#include <iostream>
using namespace std;

// creating a class with name "app"
class app {
   string a;  // declaring a string to store the name of the app
   float storage, price, boot_speed;  // declaring float variables for storage, price, and boot speed
```

```
public:  // making the functions accessible outside the class
   void name();  // function to input the name of the app
   void properties();  // function to input app properties
   void compare_boot_speed(app a1, app a2);  // function to compare boot speed between two apps
};

// inline function definition for app::name()
inline void app::name() {
   {
      cout << "Enter name of app" << endl;  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> a;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
   }
}

// inline function definition for app::properties()
inline void app::properties() {
   {
      cout << "Enter the storage of in Mb: " << a << endl;  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> storage;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)

      cout << "Enter the price of " << a << endl;  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> price;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)

      cout << "Enter the Boot_speed (in secs) of {ex:3}: " << a << endl;  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      cin >> boot_speed;  // (it is an object belongs to istream ">>" refers to bits from keyboard to memory)
   }
}

// inline function definition for app::compare_boot_speed()
inline void app::compare_boot_speed(app a1, app a2) {
   {
      if (a1.boot_speed < a2.boot_speed) {
         cout << a1.a << " is better,";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      } else if (a1.boot_speed > a2.boot_speed) {
         cout << a2.a << " is better";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      } else {
```

```
        cout << "Both are equal";  // (it is an object belongs to ostream class "<<" refers to bits are sending from memory to desktop)
      }
   }
}

int main() {
   app a1, a2, check;  // creating objects of class app

   a1.name();  // calling name function for first app
   a2.name();  // calling name function for second app

   a1.properties();  // calling properties function for first app
   a2.properties();  // calling properties function for second app

   check.compare_boot_speed(a1, a2);  // comparing boot speeds of both apps
}

==========================================================================
//5 Inline Function with Two Parameters (Pass-by-Value)
#include <iostream>
// Include the iostream library to perform input and output operations.

using namespace std;
// Use the standard namespace to avoid typing "std::" before standard library components.

class Calculator {
public:
   // Declaration of a simple Calculator class with public access.

   inline int add(int x, int y) {
   // Inline function to add two integers. The 'inline' keyword suggests that this function's definition will be copied wherever it is called, which helps in reducing the overhead of function calls.

      return x + y;
   // This returns the sum of the two integers passed as arguments.
   }
};

int main() {
   // The main function where the execution of the program begins.

   Calculator calc;
   // Create an instance of the Calculator class.

   cout << "Sum: " << calc.add(5, 10) << endl;
   // Output the result of adding 5 and 10 using the 'add' method of the Calculator object.
```

```
    // 'cout' is used to display the output to the console.

    return 0;
    // Exit the main function, indicating successful completion of the program.
}
```

==========

6.// Inline Function with Pass-by-Reference

```
#include <iostream>
// Include the input/output stream library for console input and output
using namespace std; // Use standard namespace to avoid prefixing std:: in front of standard library functions

class Updater {      // Definition of the 'Updater' class
public:
    inline void updateValue(int &x) {  // Inline function to update a variable by reference
        x += 10;  // Increment the referenced variable 'x' by 10
    }
};

int main() {
    int value = 20;  // Declare and initialize an integer variable 'value' with 20

    Updater updater;  // Create an instance of the 'Updater' class

    updater.updateValue(value);  // Pass 'value' by reference to the 'updateValue' method to update it

    cout << "Updated Value: " << value << endl;  // Output the updated value to the console

    return 0;  // Indicate successful program termination
}
```

==========

//7. Inline Function with a Default Parameter

```
#include <iostream>  // Include the iostream library for input/output operations

using namespace std;  // Use the standard namespace to avoid prefixing with 'std::'

class Multiplier {  // Define a class named Multiplier
public:
    inline int multiply(int x, int y = 2) {  // Inline function declaration with default parameter y=2
        // This function multiplies two integers, x and y.
        return x * y;  // Returns the product of x and y
    }
};
```

```
int main() {  // Main function execution starts here
   Multiplier mult;  // Create an object of the Multiplier class

   cout << "Result (default multiplier): " << mult.multiply(5) << endl;
   // Outputs the result of calling multiply with x=5, using the default value of y=2

   cout << "Result (custom multiplier): " << mult.multiply(5, 3) << endl;
   // Outputs the result of calling multiply with x=5 and y=3, as a custom multiplier is provided

   return 0;  // End of main function
}
```

==========

//8. Inline Function with String Parameter

```
#include <iostream>      // Include the input-output stream library for console I/O operations
#include <string>        // Include the string library for working with string data types
using namespace std;     // Use the standard namespace to avoid prefixing with "std::"

class Greeter {          // Declaration of a class named Greeter
public:
   inline void greet(string name) {  // Inline function to greet a user, taking a string
parameter `name`
      cout << "Hello, " << name << "!" << endl;  // Outputs a greeting message including
the user's name
   }
};

int main() {             // Main function execution begins here
   Greeter greeter;      // Create an instance of the Greeter class
   greeter.greet("Alice");  // Calls the greet function with the argument "Alice", passing the
string by value
   return 0;             // Return 0 to indicate successful execution of the program
}
```

==========

9. Inline function with return by reference

```
#include <iostream>  // Header file for input/output operations
using namespace std;  // Allows using standard library objects without prefixing with 'std::'

class Array {
   int arr[3];  // Private member variable: an array of 3 integers

public:
   inline int &getElement(int index) {  // Inline function to get a reference to an array
element
      return arr[index];  // Returns a reference to the array element at the specified index
   }
};
```

```
int main() {
    Array myArray;  // Creating an object of the Array class

    myArray.getElement(0) = 42;  // Directly modifying the first element of the array by assigning a value

    cout << "Element at index 0: " << myArray.getElement(0) << endl;  // Output the value at index 0

    return 0;  // Exit the program successfully
}
```

==========