OOP Theory Assignment 01

Question 02:

C: > Users > 3TEE > Desktop > CPP FILES > A1-Q2[23K2005].cpp > ...

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

#define N 5 // Number of tables

class Table {
    private:
        int totalSeats; // 4 or 8
        int occSeats; // Jitne log bethe hon
        int freeSeats; // Jitni seats free hon
        bool clean;

    public:
        Table () { // Default constructor
            totalSeats = 4;
            occSeats = 0;
            clean = true;
        }
        Table (int capacity) { // Parameterized constructor
            if (capacity % 4 == 0 || capacity % 8 == 0) {
                totalSeats = capacity;
            }

            else {
                int roundedCapacity = std::round((float)capacity / 4) * 4;
                if (capacity % 8 > 4) {
                    roundedCapacity = std::round((float)capacity / 8) * 8;
                }
                totalSeats = roundedCapacity;
            }
        }

        int getTotalSeats() const { return totalSeats; }
        int getOccSeats() const { return occSeats; }
        int getFreeSeats() const { return freeSeats; }
        bool isClean() const { return clean; }

        void changeTableProperties (int tempCapacity, int size) { // Called in OccupyTable
            totalSeats = tempCapacity;
            occSeats = size;
            freeSeats = totalSeats - occSeats;
        }
```

```
        void resetTable() { // Called in EmptyTable
            clean = true;
            occSeats = 0;
            freeSeats = totalSeats;
        }

        friend void Occupytable(Table tableArray[], int tableIndex, int sizeOfFriendsGroup);
        friend void Emptytable(Table tableArray[], int tableIndex);

        void useTable () {
            cout<< "The table is being used by " << occSeats << " people." << endl;
            freeSeats = totalSeats - occSeats;
        }

        void lunchOnTable () {
            cout<< "The table is now dirty, after the lunch." << endl;
            clean=0;
        }

        void leaveTable () {
            cout<<"The table is now empty and ready to be cleaned." << endl;
            freeSeats = totalSeats;
        }

        void cleanTable () {
            cout<<"The table has been cleaned and ready to allot." << endl;
            clean=1;
        }
};

void OccupyTable (Table tableArray[], int tableIndex, int sizeOfFriendsGroup) {
    int tempTotalSeats = tableArray[tableIndex].getTotalSeats();
    int tempOccSeats = tableArray[tableIndex].getOccSeats();
    bool tempClean = tableArray[tableIndex].isClean();

    if (tableIndex >= 0 && tableIndex < N) {
       if (tempTotalSeats >= sizeOfFriendsGroup) {
            cout << "You have been assigned table number " << tableIndex + 1 << "! The table has a capacity to fit " << tableArray[tableIndex].getTotalSeats() << " people." << endl;
            tableArray[tableIndex].changeTableProperties(tableArray[tableIndex].getTotalSeats(), sizeOfFriendsGroup);
        } else {
            cout << "Table " << tableIndex + 1 << " cannot be occupied." << endl;
        }
    } else {
        cout << "Invalid table number." << endl;
    }
}
```

```
void EmptyTable(Table tableArray[], int tableIndex) {
    if (tableIndex >= 0 && tableIndex < N) {
        cout<<"Table " << tableIndex + 1 << " has been made empty and ready to re-use." << endl;
        tableArray[tableIndex].resetTable();
    } else {
        cout << "Invalid table number." << endl;
    }
}

int main () {
    cout<<"********************"<<endl;
    cout<<"Name: Muhammad Hammad"<<endl;
    cout<<"Roll no: 23K-2005"<<endl;
    cout<<"********************"<<endl;

    Table tableArray[N] = {Table(4), Table(8), Table(8), Table(8), Table(4)};

    OccupyTable(tableArray, 0, 4);
    OccupyTable(tableArray, 1, 6);

    tableArray[0].useTable();
    tableArray[0].lunchOnTable();
    tableArray[0].leaveTable();
    tableArray[0].cleanTable();

    EmptyTable(tableArray, 1);

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\3TEE> cd "c:\Users\3TEE\Desktop\CPP FILES\" ; if ($?) { g++ A1-Q2-23K2005.cpp -o A1-Q2-23K2005 } ; if ($?) { .\A1-Q2-23K2005 }
********************
Name: Muhammad Hammad
Roll no: 23K-2005
********************
You have been assigned table number 1! The table has a capacity to fit 4 people.
You have been assigned table number 2! The table has a capacity to fit 8 people.
The table is being used by 4 people.
The table is now dirty, after the lunch.
The table is now empty and ready to be cleaned.
The table has been cleaned and ready to allot.
Table 2 has been made empty and ready to re-use.
PS C:\Users\3TEE\Desktop\CPP FILES>
```

Question 03:

C: > Users > 3TEE > Desktop > CPP FILES > A1-Q3-23K2005.cpp > main()

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

class ChessPiece {
private:
    string name;
    char color;
    char symbol;

public:
    ChessPiece(string name, char color, char symbol) : name(name), color(color), symbol(symbol) {}

    string getName() const { return name; }
    char getColor() const { return color; }
    char getSymbol() const { return symbol; }
};

class ChessBoard {
private:
    vector<vector<ChessPiece*>> cb;

public:
    ChessBoard() {
        cb.resize(8, vector<ChessPiece*>(8, nullptr));
        initializeWhitePieces();
        initializeBlackPieces();
    }

    void initializeWhitePieces() {
        const char pieces[] = {'r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'};
        for (int i=0; i<8; ++i)
            cb[0][i] = new ChessPiece(getPieceName(pieces[i]), 'w', pieces[i]);
        for (int i=0; i<8; ++i)
            cb[1][i] = new ChessPiece("p", 'w', 'p');
    }

    void initializeBlackPieces() {
        const char pieces[] = {'R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R'};
        for (int i=0; i<8; ++i)
            cb[7][i] = new ChessPiece(getPieceName(pieces[i]), 'b', pieces[i]);
        for (int i=0; i<8; ++i)
            cb[6][i] = new ChessPiece("P", 'b', 'P');
    }

```

```
    string getPieceName(char symbol) const {
        switch (symbol) {
            case 'r': return "Rook";
            case 'n': return "Knight";
            case 'b': return "Bishop";
            case 'q': return "Queen";
            case 'k': return "King";
            default:  return "";
        }
    }

    ~ChessBoard() {
        for (int i=0; i<8; ++i)
            for (int j=0; j<8; ++j)
                delete cb[i][j];
    }

    void display() const {
        cout<<"The chessboard is displayed below."<<endl;
        cout << "  a b c d e f g h" << endl;
        for (int i=0; i<8; ++i) {
            cout<< 8 - i << " ";
            for (int j=0; j<8; ++j) {
                if (cb[i][j] != nullptr)
                    cout<<cb[i][j]->getSymbol()<<" ";
                else
                    cout<<". ";
            }
            cout<< 8 - i << endl;
        }
        cout<< "  a b c d e f g h" << endl;
    }

    bool movePiece(string source, string destination) {
        int currRow = 8 - (source[1] - '0');
        int currCol = source[0] - 'a';
        int targetRow = 8 - (destination[1] - '0');
        int targetCol = destination[0] - 'a';

        if (currRow < 0 || currRow > 7 || currCol < 0 || currCol > 7 ||
            targetRow < 0 || targetRow > 7 || targetCol < 0 || targetCol > 7) {
            cout<<"The current or targetted position is invalid. Please try again."<<endl;
            return false;
        }
```

```
        if (cb[currRow][currCol] == nullptr) {
            cout<<"The targetted position has no piece. Please try again."<<endl;
            return false;
        }

        char pieceSymbol = cb[currRow][currCol]->getSymbol();
        cout<<"Moving piece with symbol "<<pieceSymbol<<" from "<<source<<" to "<<destination<<endl;

        delete cb[targetRow][targetCol];
        cb[targetRow][targetCol] = cb[currRow][currCol];
        cb[currRow][currCol] = nullptr;
        return true;
    }
};

int main() {
    cout<<"********************"<<endl;
    cout<<"Name: Muhammad Hammad"<<endl;
    cout<<"Roll no: 23K-2005"<<endl;
    cout<<"********************"<<endl;

    ChessBoard cb;
    cout<<"Initial Game Board"<<endl;
    cb.display();

    cout<<"----------------------------------------"<<endl;
    cb.movePiece("b8", "a6"); // Knight's move
    cb.display();
    cout<<"\n";

    cout<<"----------------------------------------"<<endl;
    cb.movePiece("b8", "d7"); // Pawn's move
    cb.display();

    return 0;
}
```

```
PS C:\Users\3TEE> cd "c:\Users\3TEE\Desktop\CPP FILES\" ; if ($?) { g++ A1-Q3-23K2005.cpp -o A1-Q3-23K2005 } ; if ($?) { .\A1-Q3-23K2005 }
**********************
Name: Muhammad Hammad
Roll no: 23K-2005
**********************
Initial Game Board
The chessboard is displayed below.
  a b c d e f g h
8 r n b q k b n r 8
7 p p p p p p p p 7
6 . . . . . . . . 6
5 . . . . . . . . 5
4 . . . . . . . . 4
3 . . . . . . . . 3
2 P P P P P P P P 2
1 R N B Q K B N R 1
  a b c d e f g h
----------------------------------------
Moving piece with symbol n from b8 to a6
The chessboard is displayed below.
  a b c d e f g h
8 r . b q k b n r 8
7 p p p p p p p p 7
6 n . . . . . . . 6
5 . . . . . . . . 5
4 . . . . . . . . 4
3 . . . . . . . . 3
2 P P P P P P P P 2
1 R N B Q K B N R 1
  a b c d e f g h

----------------------------------------
The targetted position has no piece. Please try again.
The chessboard is displayed below.
  a b c d e f g h
8 r . b q k b n r 8
7 p p p p p p p p 7
6 n . . . . . . . 6
5 . . . . . . . . 5
4 . . . . . . . . 4
3 . . . . . . . . 3
2 P P P P P P P P 2
1 R N B Q K B N R 1
  a b c d e f g h
PS C:\Users\3TEE\Desktop\CPP FILES>
```

Question 04:

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

class RollerCoaster {
    string name;
    int height;
    int length;
    float speed;
    int capacity; // amount of people that can be seated at once
    int currentNumRiders; // number of passengers/riders currently seated in the roller coaster
    bool RideInProgress;

    public:
        // Default Constructor
        RollerCoaster () {
            name = "roller coaster";
            height = 500;
            length = 2000;
            capacity = 20;
            currentNumRiders = 0;
            RideInProgress = false;
            speed = 0;
        }

        // Parameterized Constructor
        RollerCoaster (string name, int height, int length, float speed, int capacity) { // Round off wali bhi done
            this->name = name;
            this->height = height;
            this->length = length;
            this->capacity = capacity;
            this->speed = speed;
            currentNumRiders = 0;

            if (capacity <= 3) {
                this->capacity = 4;
            } else {
                if (capacity % 2 == 0 || capacity % 3 == 0) {
                    this->capacity = capacity;
                } else {
                    int nextMultipleOfTwo = ((capacity / 2) + 1) * 2;
                    int nextMultipleOfThree = ((capacity / 3) + 1) * 3;

                    if (abs(nextMultipleOfTwo - capacity) < abs(nextMultipleOfThree - capacity)) {
                        this->capacity = nextMultipleOfTwo;
                    } else {
                        this->capacity = nextMultipleOfThree;
                    }
```

```cpp
        // Setter functions
        void setProperties (string name, int height, int length, float speed, int capacity, int currentNumRiders) { // Round off wali bhi done
            this->name = name;
            this->height = height;
            this->length = length;
            this->capacity = capacity;
            this->speed = speed;
            this->currentNumRiders = currentNumRiders;

            if (capacity <= 3) {
                this->capacity = 4;
            } else {
                if (capacity % 2 == 0 || capacity % 3 == 0) {
                    this->capacity = capacity;
                } else {
                    int nextMultipleOfTwo = ((capacity / 2) + 1) * 2;
                    int nextMultipleOfThree = ((capacity / 3) + 1) * 3;

                    if (abs(nextMultipleOfTwo - capacity) < abs(nextMultipleOfThree - capacity)) {
                        this->capacity = nextMultipleOfTwo;
                    } else {
                        this->capacity = nextMultipleOfThree;
                    }
                }
            }
        }

        // Getter functions
        string getName() const { return name; }
        int getHeight() const { return height; }
        int getLength() const { return length; }
        float getSpeed() const { return speed; }
        int getCapacity() const { return capacity; }
        int getCurrentNumRiders() const { return currentNumRiders; }
        bool isRideInProgress() const { return RideInProgress; }

        void displayRideDetails () { // Extra function, to display details
            cout<<"--------- Details of the ride --------- " << endl;
            cout<<"Name: " << name << endl;
            cout<<"Height: " << height << " meters" << endl;
            cout<<"Length: " << length << " meters" << endl;
            cout<<"Speed: " << speed << " m/s" << endl;
            cout<<"Capacity: " << capacity << " passengers" << endl;
            cout<<"Current passengers/riders: " << currentNumRiders << endl;
            cout<<"----------------------------------------" << endl<<endl;
        }
```

```
int loadPassengers (int passengersToLoad) {
    int excessPassengers = 0;
    if (!RideInProgress) {
        if (capacity<=passengersToLoad) {
            currentNumRiders = capacity;
            excessPassengers = passengersToLoad - currentNumRiders;
            cout<< currentNumRiders << " passenger(s) have been loaded in the roller coaster!" << endl;
                if (excessPassengers != 0) {
                    cout<<"But " << excessPassengers << " passenger(s) were not seated due to insufficient space!" << endl;
                }
            return  excessPassengers;// returning excess number of passengers
        }
        else if (capacity>passengersToLoad) {
            currentNumRiders = passengersToLoad;
            cout<< currentNumRiders << " passengers have been loaded in the roller coaster!" << endl;
            return 0;
        }
        else {
            cout<<"Invalid passenger number." << endl;
        }
    }
    else {
        cout<<"The rider is in progress. People cannot be loaded right now." << endl;
        return 0;
    }
}

int startRide () {
    char choice;

    if (!RideInProgress) {
        if (capacity == currentNumRiders) {
            cout<<"All the seats have been occupied and the ride is ready to start." << endl;
            RideInProgress = true;
            return 0;
        }
        else if (currentNumRiders < capacity) {
            RideInProgress = false;
            int emptySeats = capacity - currentNumRiders;
            cout<< emptySeats << " seats are still vacant. Ride will not start until it's full." << endl;
            cout<<"Do you want to load " << emptySeats << " more passengers to start the ride? (y/n): " << endl;
            cin>> choice;
```

```
                switch (choice) {
                    case 'y':
                    case 'Y':
                        currentNumRiders = capacity;
                        cout<< emptySeats << " passengers have been successfully loaded and the ride is going to start!" << endl;
                        RideInProgress = true;
                    break;

                    case 'n':
                    case 'N':
                        cout<<"More passengers will not be added and the ride will not start." << endl;
                    break;

                    default:
                        cout<<"Invalid choice. Please try again." << endl;
                    break;
                }

                return emptySeats;
            }
        }
        else {
            return -1;
        }
    }

    int stopRide () {
        if (RideInProgress) {
            cout<<"The ride has been stopped." <<endl;
            RideInProgress = false;
        }
        else {
            cout<<"The ride is not moving and is already at stationary." << endl;
        }
    }

    void unloadPassengers () {
        if (!RideInProgress) {
            cout<<"The passengers have been unloaded from the roller coaster." << endl;
            currentNumRiders = 0;
        }
        else {
            cout<<"Passengers cannot be unloaded right now as the ride is in progress." << endl;
        }
    }
```

```
        void speedRide () {
            if (RideInProgress) {
                speed = 5 * speed;
                cout<<"The speed has been increased by 5 times, the speed is now " << speed << "m/s." << endl;
            }
            else {
                cout<<"The ride is not moving and therefore cannot be speeded up." << endl;
            }
        }

        void slowDownRide () {
            if (RideInProgress) {
                speed = speed / 2;
                cout<<"The speed has been decreased by 2 times, the speed is now " << speed << "m/s." << endl;
            }
            else {
                cout<<"The ride is not moving and therefore cannot be speeded down." << endl;
            }
        }
};


int main () {
    cout<<"********************"<<endl;
    cout<<"Name: Muhammad Hammad"<<endl;
    cout<<"Roll no: 23K-2005"<<endl;
    cout<<"********************"<<endl;

    string name;
    int height, length, capacity, passengersToLoad;
    float speed;

    cout<<"Enter the details of roller coaster: " << endl;
    cout<<"Name: ";
    getline(cin,name);
    cout<<"Height: "; cin>> height;
    cout<<"Length: "; cin>> length;
    cout<<"Speed: "; cin>> speed;
    cout<<"Capacity: "; cin>> capacity;
    cout<<"Passengers to load: "; cin>> passengersToLoad;
```

```
int main () {
    cout<<"********************"<<endl;
    cout<<"Name: Muhammad Hammad"<<endl;
    cout<<"Roll no: 23K-2005"<<endl;
    cout<<"********************"<<endl;

    string name;
    int height, length, capacity, passengersToLoad;
    float speed;

    cout<<"Enter the details of roller coaster: " << endl;
    cout<<"Name: ";
    getline(cin,name);
    cout<<"Height: "; cin>> height;
    cout<<"Length: "; cin>> length;
    cout<<"Speed: "; cin>> speed;
    cout<<"Capacity: "; cin>> capacity;
    cout<<"Passengers to load: "; cin>> passengersToLoad;

    RollerCoaster R1, R2(name, height, length, speed, capacity);

    cout<<endl<< "For Default Constructor" << endl;
    R1.displayRideDetails();
    cout<<"For Parameterized Constructor" << endl;
    R2.displayRideDetails();
    R2.loadPassengers(passengersToLoad);
    R2.startRide();
    R2.speedRide();
    R2.slowDownRide();
    R2.stopRide();
    R2.unloadPassengers();



    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\3TEE> cd "c:\Users\3TEE\Desktop\" ; if ($?) { g++ A1-Q4-23K2005.cpp -o A1-Q4-23K2005 } ; if ($?) { .\A1-Q4-23K2005 }
********************
Name: Muhammad Hammad
Roll no: 23K-2005
********************
Enter the details of roller coaster:
Name: Jhoola
Height: 250
Length: 250
Speed: 30
Capacity: 10
Passengers to load: 8

For Default Constructor
--------- Details of the ride ---------
Name: roller coaster
Height: 500 meters
Length: 2000 meters
Speed: 0 m/s
Capacity: 20 passengers
Current passengers/riders: 0
---------------------------------------

For Parameterized Constructor
--------- Details of the ride ---------
Name: Jhoola
Height: 250 meters
Length: 250 meters
Speed: 30 m/s
Capacity: 10 passengers
Current passengers/riders: 0
---------------------------------------

8 passengers have been loaded in the roller coaster!
2 seats are still vacant. Ride will not start until it's full.
Do you want to load 2 more passengers to start the ride? (y/n):
y
2 passengers have been successfully loaded and the ride is going to start!
The speed has been increased by 5 times, the speed is now 150m/s.
The speed has been decreased by 2 times, the speed is now 75m/s.
The ride has been stopped.
The passengers have been unloaded from the roller coaster.
PS C:\Users\3TEE\Desktop>
```

Question 05:

```cpp
#include <iostream>
#include <vector>
#include <iomanip>

using namespace std;

class Date {
    private:
        unsigned int dd, mm, yy;

    public:
        // Constructors
        Date() { }
        Date(unsigned int d, unsigned int m, unsigned int y) : dd(d), mm(m), yy(y) { }

        // Operators
        bool operator <= (const Date &other) {
            return (yy < other.yy) || (yy == other.yy && mm < other.mm) || (yy == other.yy && mm == other.mm && dd <= other.dd);
        }

        bool operator >= (const Date &other) {
            return (yy > other.yy) || (yy == other.yy && mm > other.mm) || (yy == other.yy && mm == other.mm && dd >= other.dd);
        }
};

class Order {
    public:
        int item_id;
        int quantity;
        string r_code;

        Order(int item, int q, string rc) : item_id(item), quantity(q), r_code(rc) { }
};

class Restaurant {
    private:
        static int coupons_redeemed_count;
        string restaurant_name;
        string location;
        string r_code;
        vector<string> menu_list;
        vector<int> price_list;
        vector<string> valid_coupon_codes_list;
        int bill;
```

```
    public:
        // Constructors
        Restaurant() { }
        Restaurant(string name, string loc, string code) : restaurant_name(name), location(loc), r_code(code), bill(0) { }

        // Member Functions
        void display_menu() {
            cout << "* " << restaurant_name << " Menu *" << endl;
            cout << "-----------------------" << endl;
            cout << left << setw(15) << "#  Item Name" << "Price" << endl;
            cout << "-----------------------" << endl;

            for(int i = 0; i < menu_list.size(); i++) {
                cout << i+1 << ". " << left << setw(15) << menu_list[i] << price_list[i] << endl;
            }
            cout << "-----------------------" << endl << endl;
        }

        void generate_bill(Order& order) {
            if(order.r_code == r_code) {
                int item_id = order.item_id;
                bill += order.quantity * price_list[item_id-1];
            }
        }

        void apply_discount(Order& order) {
            bill -= price_list[order.item_id-1]; // discount for current order
        }

        void addItem(string name, int price) {
            menu_list.push_back(name);
            price_list.push_back(price);
        }

        string getCode() { return r_code; }

        string getName() { return restaurant_name; }

        string getItemName(int id) {
            if(id > 0 && id <= menu_list.size())
                return menu_list[id-1];
            return "#Item Not Found#";
        }

        int getBill() { return bill; }
};
```

```
class BOGOCoupon {
    private:
        string coupon_code;
        Date valid_from;
        Date valid_until;
        string restaurant_code;

    public:
        // Constructors
        BOGOCoupon(string code, Date start, Date end, string r_code) : coupon_code(code), valid_from(start), valid_until(end), restaurant_code(r_code) { }

        // Member Functions
        bool is_valid(Date date, string r_code) {
            return ((date >= valid_from && date <= valid_until) && restaurant_code == r_code);
        }

        string getCode() { return coupon_code; }
};

class User {
    private:
        string name;
        int age;
        int mobile_number;
        vector<BOGOCoupon> coupons_list;
        vector<BOGOCoupon> redeemed_coupons_list;
        vector<Order> order_list;
        Date order_date;

    public:
        // Constructors
        User() { }
        User(string _name, int _age, int number, Date orderDate) : name(_name), age(_age), mobile_number(number), order_date(orderDate) { }

        // Member Functions
        void accumulate_coupon(BOGOCoupon& coupon) {
            coupons_list.push_back(coupon);
        }

        int has_valid_coupon(Restaurant& restaurant) {
            string r_code = restaurant.getCode();
            for(int i = 0; i < coupons_list.size(); i++) {
                if(coupons_list[i].is_valid(order_date, r_code) && !is_redeemed_before(coupons_list[i]))
                    return i;
            }
            return -1;
        }
```

```
        void redeem_coupon(Restaurant& restaurant, Order& order) {
            int couponIndex = has_valid_coupon(restaurant);

            if(couponIndex != -1) {
                redeemed_coupons_list.push_back(coupons_list[couponIndex]);
                cout << "Successfully Redeemed Coupon: " << coupons_list[couponIndex].getCode() << endl;
                coupons_list.erase(coupons_list.begin() + couponIndex); // remove the used coupon from list
                restaurant.apply_discount(order);
            }
            else {
                cout << "No valid coupons found for restaurant: " << restaurant.getName() << endl;
            }
        }

        bool is_redeemed_before(BOGOCoupon& coupon) {
            for(int i = 0; i < redeemed_coupons_list.size(); i++) {
                if(redeemed_coupons_list[i].getCode() == coupon.getCode())
                    return true;
            }
            return false;
        }

        void place_order(Restaurant& restaurant, Order order) {
            order_list.push_back(order);
            cout << "You ordered: " << restaurant.getItemName(order.item_id) << " x" << order.quantity << endl;
            restaurant.generate_bill(order);
            cout << "Current Bill for " << restaurant.getName() << ": " << restaurant.getBill() << endl;
        }
};

// Static Member Init.
int Restaurant::coupons_redeemed_count = 0;
```

```
int main() {
    Date currentDate(20, 2, 2024);

    Restaurant FH("Food Haven", "City Center", "FH");
    FH.addItem("Sushi", 10);
    FH.addItem("Pad Thai", 20);
    FH.addItem("Mango Tango", 30);

    Restaurant PB("Pixel Bites", "Cyber Street", "PB");
    PB.addItem("Binary Burger", 40);
    PB.addItem("Quantum Quinoa", 50);
    PB.addItem("Data Donuts", 60);

    BOGOCoupon c1("FH-BOGO-12345", Date(20, 1, 2024), Date(20, 2, 2024), "FH");
    BOGOCoupon c2("PB-BOGO-67890", Date(1, 2, 2024), Date(1, 3, 2024), "PB");

    User user1("Yousuf", 20, 12345, currentDate);
    user1.accumulate_coupon(c1);
    user1.accumulate_coupon(c2);

    bool exit = false;

    while(!exit) {
        cout << "--- Program Menu ---\n";
        cout << "1. Display Food Haven menu\n";
        cout << "2. Display Pixel Bites menu\n";
        cout << "3. Place an order\n";
        cout << "4. Display Bill\n";
        cout << "5. Exit\n";
        cout << "Choose an option: ";

        int choice;
        cin >> choice;

        switch (choice) {
            case 1:
                FH.display_menu();
                break;

            case 2:
                PB.display_menu();
                break;
```

```
        case 3: {
            int restaurantChoice, itemChoice, quantity;
            char couponChoice;

            cout << "Choose a restaurant (1-Food Haven OR 2-Pixel Bites): ";
            cin >> restaurantChoice;

            Restaurant& rest = (restaurantChoice == 1) ? FH : PB;

            rest.display_menu();

            cout << "Choose an item (by number): ";
            cin >> itemChoice;

            cout << "Enter quantity: ";
            cin >> quantity;

            Order order(itemChoice, quantity, rest.getCode());
            user1.place_order(rest, order);

            if(quantity > 1 && user1.has_valid_coupon(rest) != -1) {
                cout << "Would you like to use coupon for this order? (Y/N): "; cin >> couponChoice;
                if(tolower(couponChoice) == 'y') {
                    user1.redeem_coupon(rest, order);
                    cout << "Discounted Bill: " << rest.getBill();
                }
            }
            break;
        }

        case 4: {
            int restaurantChoice;
            cout << "Choose a restaurant (1-Food Haven OR 2-Pixel Bites): ";
            cin >> restaurantChoice;
            Restaurant& rest = (restaurantChoice == 1) ? FH : PB;
            cout << "Current Bill for " << rest.getName() << ": " << rest.getBill() << endl;
            break;
        }

        case 5:
            exit = true;
            break;

        default:
            cout << "Invalid choice. Please try again.\n";
    }
```

```
PS C:\Users\3TEE> cd "c:\Users\3TEE\Desktop\" ; if ($?) { g++ A1-Q5-23K2005.cpp -o A1-Q5-23K2005 } ; if ($?) { .\A1-Q5-23K2005 }
--- Program Menu ---
1. Display Food Haven menu
2. Display Pixel Bites menu
3. Place an order
4. Display Bill
5. Exit
Choose an option: 1
* Food Haven Menu *
------------------------
#  Item Name   Price
------------------------
1. Sushi          10
2. Pad Thai       20
3. Mango Tango    30
------------------------


--- Program Menu ---
1. Display Food Haven menu
2. Display Pixel Bites menu
3. Place an order
4. Display Bill
5. Exit
Choose an option: 2
* Pixel Bites Menu *
------------------------
#  Item Name   Price
------------------------
1. Binary Burger  40
2. Quantum Quinoa 50
3. Data Donuts    60
------------------------
```

```
--- Program Menu ---
1. Display Food Haven menu
2. Display Pixel Bites menu
3. Place an order
4. Display Bill
5. Exit
Choose an option: 3
Choose a restaurant (1-Food Haven OR 2-Pixel Bites): 2
* Pixel Bites Menu *
-----------------------
#  Item Name   Price
-----------------------
1. Binary Burger  40
2. Quantum Quinoa 50
3. Data Donuts    60
-----------------------

Choose an item (by number): 3
Enter quantity: 2
You ordered: Data Donuts x2
Current Bill for Pixel Bites: 120
Would you like to use coupon for this order? (Y/N): y
Successfully Redeemed Coupon: PB-BOGO-67890
Discounted Bill: 60
```

```
--- Program Menu ---
1. Display Food Haven menu
2. Display Pixel Bites menu
3. Place an order
4. Display Bill
5. Exit
Choose an option: 4
Choose a restaurant (1-Food Haven OR 2-Pixel Bites): 2
Current Bill for Pixel Bites: 60

--- Program Menu ---
1. Display Food Haven menu
2. Display Pixel Bites menu
3. Place an order
4. Display Bill
5. Exit
Choose an option: 5
```

Question #01:

C: > Users > 3TEE > Desktop > A1-Q1-23K2005.cpp > Adopter > adopterMobileNum

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

#define N 3 // Number of pets

class Pet {
    private:
        string name;
        string species;
        string healthStatus; // Healthy or Sick
        string specialSkill; // Any 1 special skill
        int hungerLevel; // Max 5 (Assuming)
        int happinessLevel; // Max 10, Min 1
        bool status; // 1 -> Available for Adoption, 0 for Not Available

    public:
        Pet () { // Default constructor
            name = "Default pet name";
            species = "Default pet specie";
            healthStatus = "Healthy";
            hungerLevel = 0;
            happinessLevel = 0;
            status = 1;
            specialSkill = "Default special skill";
        }


        Pet (string a, string b, string c, int d, int e, int f, string g) : name(a), species(b), healthStatus(c), hungerLevel(d), happinessLevel(e), status(f), specialSkill(g) { }

        void setName () {
            string x;
            cout<<"Enter Name: ";
            //cin.ignore();
            getline(cin, x);
            name = x;
        }
        void setSpecies () {
            string x;
            cout<<"Enter Specie: ";
            //cin.ignore();
            getline(cin, x);
            species = x;
        }
        void setSpecialSkill () {
            string x;
            cout<<"Enter a Special skill :";
```

0 ⚠ 0 0 Connected to Discord Ln 223, Col 30 Spaces: 4 UTF-8 CRLF {} C++

```cpp
        }
        void setSpecialSkill () {
            string x;
            cout<<"Enter a Special skill :";
            getline(cin, x);
            specialSkill = x;
        }
        void setHealthStatus () {
            string x;
            cout<<"Enter Health Status : ";
            //cin.ignore();
            getline(cin, x);
            healthStatus = x;;
        }
        void setHungerLevel () {
            int x;
            cout<<"Enter Hunger level (1-5) : ";
            cin>> x;
            hungerLevel = x;
        }
        void setHappinessLevel () {
            int x;
            cout<<"Enter Happiness level (1-10): ";
            cin>> x;
            happinessLevel = x;
        }
        void setStatus () {
            bool x;
            cout<<"Enter Adoption status (1 for Available & 0 for Not available): ";
            cin>> x;
            status =x;
            cin.ignore();
        }

        void displayPetDetails() {
            cout<<"Name: " << name << endl;
            cout<<"Specie: " << species << endl;
            cout<<"Health Status: " << healthStatus << endl;
            cout<<"Hunger Level: " << hungerLevel << endl;
            cout<<"Happiness Level: " << happinessLevel << endl;
            cout<<"Special skill: " << specialSkill << endl;
            cout<<"Adoption status: " << status << endl;
        }

```

C: > Users > 3TEE > Desktop > A1-Q1-23K2005.cpp > Adopter > adopterMobileNum

```cpp
        string getName() const { return name; }
        string getSpecies() const { return species; }
        string getHealthStatus() const { return healthStatus; }
        string getSpecialSkill() const { return specialSkill; }
        int getHungerLevel() const { return hungerLevel; }
        int getHappinessLevel() const { return happinessLevel; }
        bool getStatus() const { return status; }

        void updateHappiness () {
            int choice, tempHappinessLevel, updatedHappiness;

            if (happinessLevel >=1 && happinessLevel< 10) {
                cout<<"Choose one of the following: "  <<endl << "1. Play with the pet\n2. Cuddle the pet\n3. Take them for a walk\n4. Feed the pet\n5. Exit program";
                cin>> choice;

                switch (choice) {
                    case 1: // Base case lagani ha k agar 10 se zyada hua to kia karenge
                        if (happinessLevel+3 > 10) {
                            tempHappinessLevel = happinessLevel;
                            happinessLevel = 10 % (happinessLevel+3);
                            updatedHappiness = happinessLevel - tempHappinessLevel;
                        }
                        else {
                            tempHappinessLevel = happinessLevel;
                            happinessLevel = happinessLevel + 3;
                            updatedHappiness = happinessLevel - tempHappinessLevel;
                        }
                        cout<<"The pet is happy to play with you and its happiness level is increased by " << updatedHappiness <<endl;
                    break;

                    case 2:
                        if (happinessLevel+3 > 10) {
                            tempHappinessLevel = happinessLevel;
                            happinessLevel = 10 % (happinessLevel+3);
                            updatedHappiness = happinessLevel - tempHappinessLevel;
                        }
                        else {
                            tempHappinessLevel = happinessLevel;
                            happinessLevel = happinessLevel + 3;
                            updatedHappiness = happinessLevel - tempHappinessLevel;
                        }
                        cout<<"The pet enjoyed the cuddle and its happiness level is increased by " << updatedHappiness <<endl;
                    break;
```

```cpp
                    case 3:
                        if (happinessLevel+3 > 10) {
                            tempHappinessLevel = happinessLevel;
                            happinessLevel = 10 % (happinessLevel+3);
                            updatedHappiness = happinessLevel - tempHappinessLevel;
                        }
                        else {
                            tempHappinessLevel = happinessLevel;
                            happinessLevel = happinessLevel + 3;
                            updatedHappiness = happinessLevel - tempHappinessLevel;
                        }
                        cout<<"The pet enjoyed the walk and its happiness level is increased by " << updatedHappiness <<endl;
                    break;

                    case 4:
                        updateHealth();
                    break;

                    case 5:
                        cout<<"The program has been closed." << endl;
                        exit(1);
                    break;
                }
            }
            else {
                cout<<"The pet is already fully happy!" <<endl;
            }
        }

        void updateHealth () {
            char choice;

            cout<<"Is the pet running around and playing energetically? (y or n): ";
            cin>>choice;

            if (choice == 'y' || choice == 'Y') {
                cout<<"The pet is healthy!" << endl;
                healthStatus = "Healthy";
            }
            else if (choice=='n' || choice == 'N') {
                cout<<"The pet must be sick!" << endl;
                healthStatus = "Sick";
            }
            else {
                cout<<"Invalid choice." << endl;
            }
```

```
        void updateHunger () {
            char choice;

            if (hungerLevel >= 0 && hungerLevel <=4) {
                cout<<"The pet is hungry! Do you want to feed the pet? ";
                cin>> choice;

                if (choice == 'y' || choice == 'Y') {
                    hungerLevel = 5% (hungerLevel + 3);

                    if (happinessLevel < 10) {
                        happinessLevel = happinessLevel + 1;
                        cout<<"The pet is now feeded and happiness level is increased by 1!" << endl;
                    }
                    else {
                        cout<<"The pet is now feeded and happiness level is already maximum!" << endl;
                    }

                }

                else if (choice=='n' || choice == 'N') {
                    if (happinessLevel >= 1 && happinessLevel <= 10)
                        happinessLevel = happinessLevel - 1;
                    cout<<"No food given, the pet will remain hungry and its happiness level is decreased by 1!" << endl;
                }

                else {
                    cout<<"Invalid choice." << endl;
                }
            }
            else {
                cout<<"The pet is full already and does not want to be feeded!";
            }
        }

        //friend void adoptPet();
};

int numOfPetsToAdopt = 0, numOfPetsToReturn = 0;
int totalNumOfPets = numOfPetsToAdopt - numOfPetsToReturn; // numOfPetsToAdopt = number of pets user wants to adopt

```

```
class Adopter {
    private:
        string adopterName;
        int adopterMobileNum;
        Pet pet[N];
        string adoptedPetRecords[N][7]; // 2D array to store pet number in columns and their corresponding data in the rows. 7 because Pet class k 7 attributes hain

    public:

        // Setter function to populate the Pet's Object Array.
        // Uske bad jab user pet adopt ya return krega to us index ka corresponding data adoptedPetRecords wali string me copy/remove kr denge.
        void setPetArray () {
            for (int i=0; i<N; i++) {
                cout<<"\tEnter the details for pet " << i+1 << endl;
                pet[i].setName ();
                pet[i].setSpecies();
                pet[i].setSpecialSkill();
                pet[i].setHealthStatus();
                pet[i].setHungerLevel();
                pet[i].setHappinessLevel();
                pet[i].setStatus();
            }
        }

        int adoptPet() {
            int i, j;
            int petSelection[N];

            for (i=0; i<N; i++) { // 3 baar chalega kyuke assuming 3 pets
                cout<<"\tDetails for Pet " << i+1 << endl;
                pet[i].displayPetDetails();
            }

            cout<<"\n\nEnter your name: ";
            //cin.ignore();
            getline(cin, adopterName);
            cout<<"Enter your cell phone number: ";
            cin>>adopterMobileNum;
            cout<<"How many pets do you want to adopt?: ";
            cin >> numOfPetsToAdopt;

            if (numOfPetsToAdopt==0) {
                cout<<"You have chosen to not adopt any pet." << endl;
                return 0;
            }

```

```
266             int *ArrOfPetsToAdopt = new int(numOfPetsToAdopt); // Wo pet number, jo user ko adopt krna ha
267
268             for (i=0; i<numOfPetsToAdopt; i++) {
269                 cout<<"Enter the pet number which you want to adopt: ";
270                 cin >> ArrOfPetsToAdopt[i];
271             }
272
273             for (i=0; i<numOfPetsToAdopt; i++) {
274                 cout<<"Storing the records for Pet " << i+1 << endl;
275                     int petIndex = ArrOfPetsToAdopt[i] - 1; // Subtracting 1 to obtain the index of the pet number that user wants to adopt.
276                     adoptedPetRecords[i][0] = pet[petIndex].getName();
277                     adoptedPetRecords[i][1] = pet[petIndex].getSpecies();
278                     adoptedPetRecords[i][2] = pet[petIndex].getSpecialSkill();
279                     adoptedPetRecords[i][3] = pet[petIndex].getHealthStatus();
280                     adoptedPetRecords[i][4] = to_string(pet[petIndex].getHungerLevel());
281                     adoptedPetRecords[i][5] = to_string(pet[petIndex].getHappinessLevel());
282             }
283             return 0;
284         }
285
286         int returnPet() {
287             int i, j;
288             cout<<"How many pets do you want to return?: ";
289             cin >> numOfPetsToReturn;
290
291             if (numOfPetsToReturn==0) {
292                 cout<<"You have chosen to not return any pet." << endl;
293                 return 0;
294             }
295
296             int *petsToReturn = new int(numOfPetsToReturn); // Wo pet number, jo user ko return krna ha
297
298             for (i=0; i<numOfPetsToReturn; i++) {
299                 cout<<"Enter the pet number which you want to return: ";
300                 cin >> petsToReturn[i];
301             }
302
303             for (i = 0; i < numOfPetsToReturn; i++) {
304                 cout << "Deleting the records for Pet " << petsToReturn[i] << endl;
305
306                 // yahan par subtracting 1 to obtain the index of the pet number that user wants to delete.
307                 int petIndex = petsToReturn[i] - 1;
308
```

```
309                 for (int j = petIndex; j < numOfPetsToReturn - 1; j++) {
310                     adoptedPetRecords[j][0] = adoptedPetRecords[j + 1][0];
311                     adoptedPetRecords[j][1] = adoptedPetRecords[j + 1][1];
312                     adoptedPetRecords[j][2] = adoptedPetRecords[j + 1][2];
313                     adoptedPetRecords[j][3] = adoptedPetRecords[j + 1][3];
314                     adoptedPetRecords[j][4] = adoptedPetRecords[j + 1][4];
315                     adoptedPetRecords[j][5] = adoptedPetRecords[j + 1][5];
316                 }
317
318                 adoptedPetRecords[numOfPetsToReturn - 1][0] = "";
319                 adoptedPetRecords[numOfPetsToReturn - 1][1] = "";
320                 adoptedPetRecords[numOfPetsToReturn - 1][2] = "";
321                 adoptedPetRecords[numOfPetsToReturn - 1][3] = "";
322                 adoptedPetRecords[numOfPetsToReturn - 1][4] = "";
323                 adoptedPetRecords[numOfPetsToReturn - 1][5] = "";
324
325                 numOfPetsToReturn--;
326             }
327             cout<<"The pet has been returned and your adoption list has been successfully updated." << endl;
328             displayAdoptedPets();
329         }
330
331
332         void displayAdopterDetails() {
333             cout << "\tctively Adopted Pets:" << endl;
334             cout << "Adopter Name: " << adopterName << endl;
335             cout << "Mobile Number: " << adopterMobileNum << endl;
336             cout << "Adopted Pets:" << endl;
337         }
338
339         void displayAdoptedPets () {
340             for (int i = 0; i < totalNumOfPets; ++i) {
341                 cout<<"Pet's Name: " << adoptedPetRecords[i][0] << endl;
342                 cout<<"Pet's Species: " << adoptedPetRecords[i][1] << endl;
343                 cout<<"Pet's special skill: " << adoptedPetRecords[i][2] << endl;
344                 cout<<"Pet's Health Status: " << adoptedPetRecords[i][3] << endl;
345                 cout<<"Pet's Hunger Level: " << adoptedPetRecords[i][4] << endl;
346                 cout<<"Pet's Happiness Level: " << adoptedPetRecords[i][5] << endl;
347             }
348         }
349 };
350
```

```cpp
int main() {
    cout<<"********************"<<endl;
    cout<<"Name: Muhammad Hammad"<<endl;
    cout<<"Roll no: 23K-2005"<<endl;
    cout<<"********************"<<endl;

    // Create pets using default and parameterized constructors
    Pet Pet1; // Using default constructor
    Pet Pet2("Buddy", "Dog", "Healthy", 2, 8, 1, "Catch a ball"); // Using parameterized constructor

    // Displaying details of pets
    cout << "Details of Default Pet:" << endl;
    Pet1.displayPetDetails();
    cout << endl;

    cout << "Details of Parameterized Pet:" << endl;
    Pet2.displayPetDetails();
    cout << endl;

    // Adopter object bana rahe hain aur usse pets adopt/return krwa rahe hain
    Adopter adopter;
    adopter.setPetArray();

    adopter.adoptPet();
    adopter.displayAdopterDetails();
    adopter.displayAdoptedPets();
    adopter.returnPet();

    // Interact with pets
    Pet2.updateHappiness();
    Pet2.updateHealth();
    Pet2.updateHunger();

    return 0;
}
```

```
********************
Name: Muhammad Hammad
Roll no: 23K-2005
********************
Details of Default Pet:
Name: Default pet name
Specie: Default pet specie
Health Status: Healthy
Hunger Level: 0
Happiness Level: 0
Special skill: Default special skill
Adoption status: 1

Details of Parameterized Pet:
Name: Buddy
Specie: Dog
Health Status: Healthy
Hunger Level: 2
Happiness Level: 8
Special skill: Catch a ball
Adoption status: 1

        Enter the details for pet 1
Enter Name: Ann
Enter Specie: Cat
Enter a Special skill :Meows
Enter Health Status : Healthy
Enter Hunger level (1-5) : 2
Enter Happiness level (1-10): 5
Enter Adoption status (1 for Available & 0 for Not available): 1
        Enter the details for pet 2
Enter Name: Tommy
Enter Specie: Dog
Enter a Special skill :Runs fast
Enter Health Status : Healthy
Enter Hunger level (1-5) : 4
Enter Happiness level (1-10): 7
Enter Adoption status (1 for Available & 0 for Not available): 1
        Enter the details for pet 3
Enter Name: Ovi
Enter Specie: Parrot
Enter a Special skill :Speaking
Enter Health Status : Sick
Enter Hunger level (1-5) : 2
Enter Happiness level (1-10): 3
```

```
        Enter the details for pet 3
Enter Name: Ovi
Enter Specie: Parrot
Enter a Special skill :Speaking
Enter Health Status : Sick
Enter Hunger level (1-5) : 2
Enter Happiness level (1-10): 3
Enter Adoption status (1 for Available & 0 for Not available): 1
        Details for Pet 1
Name: Ann
Specie: Cat
Health Status: Healthy
Hunger Level: 2
Happiness Level: 5
Special skill: Meows
Adoption status: 1
        Details for Pet 2
Name: Tommy
Specie: Dog
Health Status: Healthy
Hunger Level: 4
Happiness Level: 7
Special skill: Runs fast
Adoption status: 1
        Details for Pet 3
Name: Ovi
Specie: Parrot
Health Status: Sick
Hunger Level: 2
Happiness Level: 3
Special skill: Speaking
Adoption status: 1


Enter your name: Muhammad Hammad
Enter your cell phone number: 33333333
How many pets do you want to adopt?: 2
Enter the pet number which you want to adopt: 1
Enter the pet number which you want to adopt: 2
Storing the records for Pet 1
Storing the records for Pet 2
        ctively Adopted Pets:
Adopter Name: Muhammad Hammad
Mobile Number: 33333333
Adopted Pets:
How many pets do you want to return?: 1
```

```
Adopter Name: Muhammad Hammad
Mobile Number: 33333333
Adopted Pets:
How many pets do you want to return?: 1
Enter the pet number which you want to return: 2
Deleting the records for Pet 2
The pet has been returned and your adoption list has been successfully updated.
Choose one of the following:
1. Play with the pet
2. Cuddle the pet
3. Take them for a walk
4. Feed the pet
5. Exit program1
The pet is happy to play with you and its happiness level is increased by 2
Is the pet running around and playing energetically? (y or n): y
The pet is healthy!
The pet is hungry! Do you want to feed the pet? y
The pet is now feeded and happiness level is already maximum!
PS C:\Users\3TEE\Desktop>
```