```
/*
   ASSIGNMENT 3                QUESTION 1
   Write a function swap(a, b) to interchange the values of two variables.
   Do not use pointers.

   NAME : Anirudh Modi          ROLL : 002310501031        DATE : 12/09/2024
*/


#include <iostream> // For input and output operations
using namespace std;

// Global variable declaration
int number_1;  // Variable to store the first number
int number_2;  // Variable to store the second number

// Function to swap the values of number_1 and number_2
void swap() {
   int temp = number_1; // Store the value of number_1 in a temporary variable
   number_1 = number_2; // Assign the value of number_2 to number_1
   number_2 = temp;     // Assign the value stored in temp to number_2
}

int main() {
   // Prompt the user to enter the first number
   cout << "Enter the first number: ";
   cin >> number_1;  // Read the first number from user input

   // Prompt the user to enter the second number
   cout << "Enter the second number: ";
   cin >> number_2;  // Read the second number from user input

   // Call the swap function to swap the values of number_1 and number_2
   swap();

   // Output the swapped values of number_1 and number_2
   cout << "After swapping, the first number is " << number_1
        << " and the second number is " << number_2 << endl;

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

```
Enter the first number: 1
Enter the second number: 2
After swapping, the first number is 2 and the second number is 1
```

```
/*
        ASSIGNMENT 3                * QUESTION 2
        Write a function max(a, b) that will return the reference
        of the larger value. Store the returned information to x where
        x is a i) variable of type a or b ii) variable referring to
        type of a or b. In both cases modify x.
        Check also the values of a and b.

        NAME: Anirudh Modi     ROLL : 00231050131        DATE:12/09/2024
*/

#include <iostream>  // Include for input/output operations
using namespace std;
// Function to return reference of the larger value between two integers
int &max(int &a, int &b) {
    return (a > b) ? a : b;
}

int main() {
    int number_1, number_2;

    // Input the values for a and b from the user
    cout<<"Enter the first number : ";
    cin >> number_1;
    cout<<"Enter the second number : ";
    cin>>number_2;

    // x will refer to either a or b depending on which is larger.
    int &x = max(a, b);
    // Display the values before modifying x
    cout << "Values before modification" << endl;
    cout << "a=" << number_1 << "  b=" << number_2 << "  x=" << x << endl;
    // Modify the value of x, which in turn modifies the value of the larger variable
(either a or b)
    x += 5;
    // Display the values after modifying x
    cout << "Values after modification" << endl;
    cout << "a=" << number_1 << "  b=" << number_2 << "  x=" << x << endl;
    return 0;
}
```

```
Enter the first number : 1
Enter the second number : 2
Values before modification
a=1  b=2  x=2
Values after modification
a=1  b=7  x=7
```

```
/*
        ASSIGNMENT 3                * QUESTION 3
        Write a function that will have income and tax rate as arguments
        and will return tax amount. In case tax rate is not provided it
        will be automatically taken as 10%. Call it with and without
        tax rates.

        NAME: Anirudh Modi     ROLL:00231050131     DATE:12/09/2024
*/

#include <iostream>  // Include for input/output operations
using namespace std;

// Function to calculate the tax amount
// The second parameter, 'rate', has a default value of 10.0 (i.e., 10%)
double calculate(double income, double rate = 10.0) {
    // Calculate the tax by multiplying the income by the tax rate and dividing by
100
    return income * rate / 100.0;
}

int main() {
    double income, rate;  // Declare variables for income and rate

    // Prompt the user to enter the income and tax rate
    cout << "Enter the income and tax rate\n";
    cin >> income >> rate;  // Read the income and rate from user input

    // Call the calculate function without providing a rate (uses default 10%)
    cout << "When the rate is not provided, tax = " << calculate(income) << endl;

    // Call the calculate function with the rate provided by the user
    cout << "When the rate is provided, tax = " << calculate(income, rate) << endl;

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

```
Enter the income and tax rate
200000
25
When the rate is not provided, tax = 20000
When the rate is provided, amount = 50000
```

```
/*
ASSIGNMENT 3              * QUESTION 4
 Write a function void f(int) that prints "inside f(int)". Call
 the function with actual argument of type: i) int, ii) char,
 iii) float and iv) double. Add one or more function f(float)
 that prints "inside f(float)". Repeat the calls again and
 observe the outcomes.

 NAME: Anirudh Modi     ROLL:00231050131     DATE:12/09/2024
*/
#include <bits/stdc++.h>  // Header file that includes most standard libraries
using namespace std;

// Function that accepts an integer argument
void f(int a) {
    cout << "inside f(int)" << endl;
}
// Function that accepts a float argument
void f(float a) {
    cout << "inside f(float)" << endl;
}
int main() {
    // Declare variables of different types
    int var_integer = 5;
    char var_character = 'a';
    double var_double = 5.9;
    float var_float = 90.28;
    // Call function f with different types of arguments
    f(var_integer);    // Calls f(int)
    f(var_character);  // Calls f(int) because char is implicitly promoted to int
    f(var_double);     // Calls f(int) because there is no f(double), double is promoted
to int
    f(var_float);      // Calls f(float)
    return 0;
}
```

```
ERROR!
/tmp/04ia5p9jyD.cpp: In function 'int main()':
/tmp/04ia5p9jyD.cpp:30:6: error: call of overloaded 'f(double&)' is
    ambiguous
   30 |     f(var_double);
      |     ~^~~~~~~~~~~~
/tmp/04ia5p9jyD.cpp:15:6: note: candidate: 'void f(int)'
   15 | void f(int a){
      |      ^
/tmp/04ia5p9jyD.cpp:19:6: note: candidate: 'void f(float)'
   19 | void f(float a){
      |      ^
```

```
/*
        ASSIGNMENT 3                * QUESTION 5
        Define functions f(int, int) and f(char, int). Call the
        function with arguments of type (int, char), (char, char)
        and (float, float). Observe and analyze the outcome.

        NAME: Anirudh Modi     ROLL:00231050131     DATE:12/09/2024
*/

#include <bits/stdc++.h>  // Include most standard libraries
using namespace std;

// Function that accepts two integer arguments
void f(int, int) {
    cout << "inside f(int, int)" << endl;
}

// Function that accepts a char and an integer as arguments
void f(char, int) {
    cout << "inside f(char, int)" << endl;
}

int main() {
    int var_int = 5;
    char var_char = 'a';
    float var_float = 78.29;

    // Call f(int, char) - char will be promoted to int, so f(int, int) is called
    f(var_int, var_char);  // This will call f(int, int)

    // Call f(char, char) - The second char will be promoted to int, so f(char, int) is
called
    f(var_char, var_char);  // This will call f(char, int)

    // Commented out: Call f(float, float) - no matching function f(float, float), will
cause a compilation error
    // f(var_float, var_float);

    return 0;
}
```

```
inside f(int, int)
inside f(char, int)
```

```
/*
        ASSIGNMENT 3                * QUESTION 6
        Define a structure student with roll and score as attributes
        and with two member functions to take input and to show the
        data. Use the member functions to take data for a structure
        variable and to show. Write a global function i)to modify
        score and ii)to show the data again.

        NAME: Anirudh Modi     ROLL:00231050131     DATE:03/10/2024
*/

#include <bits/stdc++.h>  // Include standard library header files
using namespace std;

// Define the structure Student
struct Student {
    int roll;          // Student roll number
    float score;       // Student score

    // Member function to take input from the user
    void input() {
        cout << "Enter roll number" << endl;
        cin >> roll;  // Input the roll number
        cout << "Enter score" << endl;
        cin >> score;  // Input the score
    }

    // Member function to display student data
    void show() const {
        cout << "Roll : " << roll << endl << "Score : " << score << endl;
    }
};

// Global function to modify the score of a student
void modifyScore(Student &s, float newScore) {
    s.score = newScore;  // Update the student's score
}

// Global function to display the updated data of a student
void showData(const Student &s) {
    cout << "Updated Data :" << endl;
    cout << "Roll : " << s.roll << endl << "Score : " << s.score << endl;
}

int main() {
    Student S;  // Declare a Student structure variable
    S.input();  // Take input for the student
```

```
    S.show();   // Show the initial data of the student

    float score;  // Variable to store the new score
    cout << "Enter the score to be updated :\n";
    cin >> score;  // Input the new score

    modifyScore(S, score);  // Modify the student's score
    showData(S);            // Show the updated student data

    return 0;  // End of the program
}
```

```
Enter roll number
24
Enter score
95
Roll : 24
Score : 95
Enter the score to be updated :
76
Updated Data :
Roll : 24
Score : 76
```

```
/*
        ASSIGNMENT 3                * QUESTION 7
        Design a class TIME which stores hour, minute and second.
        The class should have the methods to support the following:
        User may give the time value in 24-hour format.
        User may give the time value in AM/PM format
        Display the time in 24-hour format.
        Display the time in AM/PM format.
        User may like to add minute with a time value.

        NAME: Anirudh Modi    ROLL:00231050131    DATE:03/10/2024
*/

#include<bits/stdc++.h>  // Include standard libraries
using namespace std;

class TIME {
    int hour;     // Stores the hour
    int minute;   // Stores the minute
    int second;   // Stores the second
public:
    // Constructor to initialize time (defaults to 0:0:0)
    TIME(int h = 0, int m = 0, int s = 0) {
        hour = h;
        minute = m;
        second = s;
    }

    // Method to set time in 24-hour format
    void seTime24(int h, int m, int s) {
        hour = h;
        minute = m;
        second = s;
    }

    // Method to set time in AM/PM format
    void seTimeAMPM(int h, int m, int s, string am_pm) {
        if(am_pm == "PM" && h != 12) {
            hour = h + 12;  // Convert PM hours to 24-hour format
        } else if(am_pm == "AM" && h == 12) {
            hour = 0;  // Convert 12 AM to 0 hours in 24-hour format
        } else {
            hour = h;  // For AM hours and 12 PM, no changes needed
        }
        minute = m;
        second = s;
    }
```

```
    // Method to display time in 24-hour format
    void display24Hour() const {
        // Displays time in HH:MM:SS format, padded with zeros for single digits
        cout << setw(2) << setfill('0') << hour << ":"
             << setw(2) << setfill('0') << minute << ":"
             << setw(2) << setfill('0') << second << "\n";
    }

    // Method to display time in AM/PM format
    void displayAMPM() const {
        int displayHours = hour;  // Copy hour to adjust for 12-hour format
        string am_pm = "AM";      // Default to AM
        if(hour == 0) {
            displayHours = 12;  // 12 AM is displayed as 12:00:00 AM
        } else if(hour == 12) {
            am_pm = "PM";  // 12 PM is displayed as 12:00:00 PM
        } else if(hour > 12) {
            displayHours = hour - 12;  // Convert 24-hour time to 12-hour time
            am_pm = "PM";  // Set PM for hours greater than 12
        }
        // Display time in HH:MM:SS AM/PM format
        cout << setw(2) << setfill('0') << displayHours << ":"
             << setw(2) << setfill('0') << minute << ":"
             << setw(2) << setfill('0') << second << " " << am_pm << "\n";
    }

    // Method to add minutes to the current time
    void addMinutes(int minutesToAdd) {
        minute += minutesToAdd;     // Add the minutes
        hour += minute / 60;        // Convert minutes to hours
        minute %= 60;               // Remainder is the new minutes
        hour %= 24;                 // Keep hours in 24-hour format
    }
};

int main() {
    int choice, hours, minutes, seconds, extra_minutes;
    string am_pm;
    TIME time1;  // Create TIME object for 24-hour format
    TIME time2;  // Create TIME object for AM/PM format

    cout << "Press 1 for time in 24 hour format\nPress 2 for time in AMPM
format\n";
    cin >> choice;

    switch(choice) {
```

```
        case 1:
            cout << "Enter the hour, minutes, seconds respectively\n";
            cin >> hours >> minutes >> seconds;
            time1.seTime24(hours, minutes, seconds);  // Set time in 24-hour format
            time1.display24Hour();                    // Display time in 24-hour format

            cout << "Enter the minutes you want to add\n";
            cin >> extra_minutes;
            time1.addMinutes(extra_minutes);          // Add minutes
            time1.display24Hour();                    // Display updated time
            break;

        case 2:
            cout << "Enter the hour, minutes, seconds and also AM or PM
respectively\n";
            cin >> hours >> minutes >> seconds >> am_pm;
            time2.seTimeAMPM(hours, minutes, seconds, am_pm);  // Set time in
AM/PM format
            time2.displayAMPM();                      // Display time in AM/PM format

            cout << "Enter the minutes you want to add\n";
            cin >> extra_minutes;
            time2.addMinutes(extra_minutes);          // Add minutes
            time2.displayAMPM();                      // Display updated time in AM/PM
format
            break;

        default:
            cout << "WRONG INPUT\n";  // Handle invalid input
    }

    return 0;
}
```

```
Press 1 for time in 24 hour format
Press 2 for time in AMPM format
1
Enter the hour, minutes, seconds respectively
23
12
25
23:12:25
Enter the minutes you want to add
100
00:52:25
```

```
Press 1 for time in 24 hour format
Press 2 for time in AMPM format
2
Enter the hour, minutes, seconds and also AM_OR_PM respectively
11
28
25
AM
11:28:25 AM
Enter the minutes you want to add
100
01:08:25 PM
```

```
/*
      ASSIGNMENT 3              * QUESTION 8

      Create a STACK class with operation for initialization, push and pop. Support
      for checking underflow and overflow conditions are also provided.

      NAME: Anirudh Modi    ROLL:00231050131    DATE:03/10/2024
*/

#include <iostream>
using namespace std;
class STACK {
   int *stackArray;   // Dynamic array to hold stack elements
   int top;           // Index of the top element
   int capacity;      // Maximum capacity of the stack
public:
   // Constructor to initialise the stack with a given capacity
   STACK(int size) {
      capacity = size;         // Set the stack capacity
      stackArray = new int[capacity];  // Allocate memory for the stack
      top = -1;                // Initialize top to -1, meaning the stack is empty
   }
   // Destructor to free up dynamically allocated memory
   ~STACK() {
      delete[] stackArray;  // Free the memory allocated for stack
   }
   // Method to check if the stack is full (overflow condition)
   bool isFull() const {
      return (top == capacity - 1);  // Stack is full if top is at the last index
   }
   // Method to check if the stack is empty (underflow condition)
   bool isEmpty() const {
      return (top == -1);  // Stack is empty if top is -1
   }
   // Method to push an element into the stack
   void push(int value) {
      if (isFull()) {
         cout << "Stack Overflow! Cannot push " << value << " into the stack.\n";
      } else {
         stackArray[++top] = value;  // Increment top and insert value
         cout << value << " pushed into the stack.\n";
      }
   }
   // Method to pop an element from the stack
   void pop() {
      if (isEmpty()) {
         cout << "Stack Underflow! Cannot pop from the stack.\n";
```

```
        } else {
            cout << stackArray[top--] << " popped from the stack.\n";  // Print and
decrement top
        }
    }
    // Method to display the current top element of the stack
    void peek() const {
        if (isEmpty()) {
            cout << "Stack is empty! No elements to display.\n";
        } else {
            cout << "Top element is: " << stackArray[top] << endl;
        }
    }
};
int main() {
    int size, choice, value;
    // Get the size of the stack from the user
    cout << "Enter the size of the stack: ";
    cin >> size;
    STACK myStack(size);  // Create a stack object
    do {
        // Display menu options
        cout << "\nStack Operations Menu:\n";
        cout << "1. Push\n";
        cout << "2. Pop\n";
        cout << "3. Peek (View top element)\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                // Push operation
                cout << "Enter a value to push into the stack: ";
                cin >> value;
                myStack.push(value);
                break;
            case 2:
                // Pop operation
                myStack.pop();
                break;
            case 3:
                // Peek operation to display the top element
                myStack.peek();
                break;
            case 4:
                // Exit option
                cout << "Exiting...\n";
```

```
            break;
        default:
            cout << "Invalid choice! Please select a valid option.\n";
    }
  } while (choice != 4);
  return 0;
}
```

```
Enter the size of the stack: 4

Stack Operations Menu:
1. Push
2. Pop
3. Peek (View top element)
4. Exit
Enter your choice: 1
Enter a value to push into the stack: 4
4 pushed into the stack.

Stack Operations Menu:
1. Push
2. Pop
3. Peek (View top element)
4. Exit
Enter your choice: 3
Top element is: 4

Stack Operations Menu:
1. Push
2. Pop
3. Peek (View top element)
4. Exit
Enter your choice: 1
Enter a value to push into the stack: 6
6 pushed into the stack.

Stack Operations Menu:
1. Push
2. Pop
3. Peek (View top element)
4. Exit
Enter your choice: 2
6 popped from the stack.

Stack Operations Menu:
1. Push
2. Pop
3. Peek (View top element)
4. Exit
Enter your choice: 4
Exiting...
```

```
/*
    ASSIGNMENT 3              * QUESTION 9

    Create an APPLICANT class with application id(auto generated as last id +1),
    name and score. Support must be there to receive applicant data, show
    applicant details and to find out number of applicants.

    NAME: Anirudh Modi    ROLL:00231050131    DATE:03/10/2024
*/

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

class APPLICANT {
   static int lastID;  // Static variable to keep track of the last application ID
   int appID;          // Unique application ID for each applicant
   string name;        // Name of the applicant
   float score;        // Score of the applicant

public:
   // Constructor to initialize applicant data
   APPLICANT(string n = "", float s = 0.0) {
      appID = ++lastID;  // Auto-generate application ID (lastID + 1)
      name = n;
      score = s;
   }

   // Method to receive applicant data
   void inputApplicant() {
      cout << "Enter applicant's name: ";
      getline(cin, name);
      cout << "Enter applicant's score: ";
      cin >> score;
      cin.ignore();  // Ignore the newline character left in the buffer
   }

   // Method to show applicant details
   void showApplicant() const {
      cout << "Application ID: " << appID << endl;
      cout << "Name: " << name << endl;
      cout << "Score: " << score << endl;
   }

   // Static method to get the number of applicants
   static int getNumberOfApplicants() {
      return lastID;
```

```
    }
};

// Initialize the static member variable
int APPLICANT::lastID = 0;

int main() {
    int choice;
    string name;
    float score;
    APPLICANT applicants[100];  // Array to store up to 100 applicants
    int totalApplicants = 0;

    do {
        cout << "\nAPPLICANT SYSTEM MENU\n";
        cout << "1. Add Applicant\n";
        cout << "2. Show Applicant Details\n";
        cout << "3. Show Total Number of Applicants\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore();  // Ignore newline

        switch (choice) {
            case 1:
                if (totalApplicants < 100) {
                    cout << "Adding a new applicant...\n";
                    applicants[totalApplicants].inputApplicant();
                    totalApplicants++;
                } else {
                    cout << "Maximum applicant limit reached!\n";
                }
                break;

            case 2:
                if (totalApplicants > 0) {
                    for (int i = 0; i < totalApplicants; i++) {
                        cout << "\nApplicant " << (i + 1) << " Details:\n";
                        applicants[i].showApplicant();
                    }
                } else {
                    cout << "No applicants to display.\n";
                }
                break;

            case 3:
```

```
            cout << "Total number of applicants: " <<
APPLICANT::getNumberOfApplicants() << endl;
            break;

        case 4:
            cout << "Exiting...\n";
            break;

        default:
            cout << "Invalid choice! Please select a valid option.\n";
      }
   } while (choice != 4);

   return 0;
}
```

```
APPLICANT SYSTEM MENU
1. Add Applicant
2. Show Applicant Details
3. Show Total Number of Applicants
4. Exit
Enter your choice: 1
Adding a new applicant...
Enter applicant's name: John Doe
Enter applicant's score: 85.5

APPLICANT SYSTEM MENU
1. Add Applicant
2. Show Applicant Details
3. Show Total Number of Applicants
4. Exit
Enter your choice: 2

Applicant 1 Details:
Application ID: 1
Name: John Doe
Score: 85.5

APPLICANT SYSTEM MENU
1. Add Applicant
2. Show Applicant Details
3. Show Total Number of Applicants
4. Exit
Enter your choice: 3
Total number of applicants: 1
```

```
/*
      ASSIGNMENT 3                * QUESTION 10

Design a STUDENT class to store roll, name, course, admission data and marks in
5 subjects. Provide methods corresponding to admission(marks are not available
then), receiving marks and preparing mark sheets. Support must be there to show
the number of students who have taken admission.

      NAME: Anirudh Modi     ROLL:00231050131     DATE:03/10/2024
*/

#include <iostream>
#include <string>
using namespace std;
class STUDENT
{
   int roll;
   string name;
   string course;
   string admissionDate;
   int marks[5];
   bool isAdmitted;
         int totalMarks;
   static int admittedCount;
public:
   STUDENT(int roll, string name, string course, string admissionDate)
   {
      this->roll = roll;
      this->name = name;
      this->course = course;
      this->admissionDate = admissionDate;
      this->isAdmitted = false;
   }
   void admitStudent()
   {
      isAdmitted = true;
      cout << "Student with Roll Number " << roll << " has been admitted.\n"
           << endl;
      admittedCount++;
   }
   void receiveMarks(int m1, int m2, int m3, int m4, int m5)
   {
      if (isAdmitted)
      {
         marks[0] = m1;
         marks[1] = m2;
         marks[2] = m3;
```

```
            marks[3] = m4;
            marks[4] = m5;
                        totalMarks = m1+m2+m3+m4+m5;
            cout << "Marks received for Roll Number " << roll << ".\n"
                << endl;
        }
        else
        {
            cout << "Student with Roll Number " << roll << " is not admitted yet.\n"
                << endl;
        }
    }
    void prepareMarkSheet()
    {
        if (isAdmitted)
        {
            cout << "Mark Sheet for Roll Number " << roll << ":" << endl;
            cout << "Name: " << name << endl;
            cout << "Course: " << course << endl;
            cout << "Admission Date: " << admissionDate << endl;
            cout << "Marks in 5 Subjects:" << endl;
            for (int i = 0; i < 5; i++)
            {
                cout << "Subject " << i + 1 << ": " << marks[i] << endl;
            }
                        cout<<"Total Marks = "<<totalMarks<<endl;
        }
        else
        {
            cout << "Student with Roll Number " << roll << " is not admitted yet.\n"
                << endl;
        }
    }
    static int getCountOfAdmittedStudents()
    {
        return admittedCount;
    }
};
int STUDENT::admittedCount = 0;
int main()
{
    STUDENT student1(101, "Anirudh Modi", "Computer Science and Engineering",
"03/10/2024");
    STUDENT student2(102, "Snehasis Mondol", "Electrical Engineering",
"05/10/2024");

    student1.admitStudent();
```

```
    student2.admitStudent();

    student1.receiveMarks(85, 90, 78, 92, 88);
    student2.receiveMarks(75, 88, 92, 84, 79);

    student1.prepareMarkSheet();
    cout << endl;
    student2.prepareMarkSheet();
    cout << endl;

    int admittedCount = STUDENT::getCountOfAdmittedStudents();
    cout << "Total number of admitted students: " << admittedCount << endl;

    return 0;
}
```

```
Student with Roll Number 102 has been admitted.

Marks received for Roll Number 101.

Marks received for Roll Number 102.

Mark Sheet for Roll Number 101:
Name: Anirudh Modi
Course: Computer Science and Engineering
Admission Date: 03/10/2024
Marks in 5 Subjects:
Subject 1: 85
Subject 2: 90
Subject 3: 78
Subject 4: 92
Subject 5: 88
Total Marks = 433

Mark Sheet for Roll Number 102:
Name: Snehasis Mondol
Course: Electrical Engineering
Admission Date: 05/10/2024
Marks in 5 Subjects:
Subject 1: 75
Subject 2: 88
Subject 3: 92
Subject 4: 84
Subject 5: 79
Total Marks = 418

Total number of admitted students: 2
```

```
/*
      ASSIGNMENT 3              * QUESTION 11

Create a class for Linked List. Consider a separate class NODE for basic node
activities and use it in class for linked lists.

      NAME: Anirudh Modi    ROLL:00231050131    DATE:03/10/2024
*/

#include <iostream>
using namespace std;

class Node {
public:
    int data;           // Data part of the node
    Node* next;         // Pointer to the next node

    // Constructor to initialize a new node
    Node(int value) {
        data = value;
        next = nullptr;
    }
};

class LinkedList {
private:
    Node* head;         // Pointer to the head node of the list

public:
    // Constructor to initialize an empty linked list
    LinkedList() {
        head = nullptr;
    }

    // Destructor to delete the linked list
    ~LinkedList() {
        Node* current = head;
        Node* nextNode;
        while (current != nullptr) {
            nextNode = current->next;
            delete current;
            current = nextNode;
        }
    }

    // Function to insert a new node at the end of the list
    void append(int value) {
```

```
    Node* newNode = new Node(value);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Function to insert a new node at the beginning of the list
void prepend(int value) {
    Node* newNode = new Node(value);
    newNode->next = head;
    head = newNode;
}

// Function to delete a node by value
void deleteNode(int value) {
    if (head == nullptr) return;

    if (head->data == value) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }

    Node* temp = head;
    while (temp->next != nullptr && temp->next->data != value) {
        temp = temp->next;
    }

    if (temp->next == nullptr) return;

    Node* nodeToDelete = temp->next;
    temp->next = temp->next->next;
    delete nodeToDelete;
}

// Function to display the list
void display() const {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
```

```
            temp = temp->next;
        }
        cout << "NULL" << endl;
    }
};

int main() {
    LinkedList list;

    list.append(10);
    list.append(20);
    list.append(30);
    list.prepend(5);

    cout << "Linked List: ";
    list.display();

    list.deleteNode(20);

    cout << "After deleting 20: ";
    list.display();

    return 0;
}
```

```
Linked List: 5 -> 10 -> 20 -> 30 -> NULL
After deleting 20: 5 -> 10 -> 30 -> NULL
```

```
/*
    ASSIGNMENT 3            * QUESTION 12

Design the class(es) for the follwoing scenario:
        -> An item list contains item code, name, rate and quantity for several items.
        -> Whenever a new item is added in the list uniqueness of item code is to be
checked.
        -> Time to time rate of the items may change.
        -> Whenever an item is issued or recived  existence of the item is checked
and quantity is updated.
        -> In case of issue, availability of quantity is also to be checked.
        -> User may also like to know the price/ quantity available for an item.

    NAME: Anirudh Modi    ROLL:00231050131    DATE:03/10/2024
*/

#include<iostream>
using namespace std;

class item
{
        int     code;
        string name;
        double rate;
        int quantity;
public:

        item(){}
        item(int code, string name, double rate, int quantity)
        {
                this->code=code;
                this->name=name;
                this->rate=rate;
                this->quantity=quantity;
        }

        int getCode()
        {
                return code;
        }

        string getName()
        {
                return name;
        }

        double getRate()
```

```
	{
		return rate;
	}

	int getQuantity()
	{
		return quantity;
	}

	void rateChange(double newRate)
	{
		rate=newRate;
	}

	void changeQuantity(int change)
	{
		quantity+=change;
		if(quantity<0)
			quantity=0;
	}
};


class itemList
{
	static const int maxTerms=100;
	int itemCount;
	item items[maxTerms];

public:
	itemList()
	{
		itemCount=0;
	}

	void addItem(int code, string name, double rate, int quantity)
	{
		if(itemCount<maxTerms)
		{
			bool exists=false;
			for(int i=0;i<itemCount;i++)
			{
				if(items[i].getCode()==code)
				{
					exists=true;
					break;
				}
```

```
                        }

                        if(exists)
                                cout<<"The item with the same code already
exists"<<endl;
                        if(!exists)
                        {
                                item temp(code,name,rate,quantity);
                                items[itemCount++]=temp;
                        }
                }

                else
                        cout<<"Item list is full."<<endl;
        }
        void updateRate(int code,double newRate)
        {
                for(int i=0;i<itemCount;i++)
                {
                        if(items[i].getCode()==code)
                        {
                                items[i].rateChange(newRate);
                                return;
                        }
                }
                cout<<"Item with the entered code not found."<<endl;
        }
        void updateQuantity(int code,int change)
        {
                for(int i=0;i<itemCount;i++)
                {
                        if(items[i].getCode()==code)
                        {
                                items[i].changeQuantity(change);
                                return;
                        }
                }
                cout<<"Item with the entered code not found."<<endl;
        };
        void getData(int code)
        {
                for(int i=0;i<itemCount;i++)
                {
                        if(items[i].getCode()==code)
                        {
                                cout<<"Item Code: "<<code<<endl;
                                cout<<"Item Name: "<<items[i].getName()<<endl;
```

```
                        cout<<"Item Rate: "<<items[i].getRate()<<endl;
                        cout<<"Item Quantity: "<<items[i].getQuantity()<<endl;
                        return;
                    }
                }
                cout<<"Item with the entered code not found."<<endl;
        }
};
int main()
{
        itemList list;
        list.addItem(101,"biscuit",20,50);
        list.addItem(102,"tea",234,38);
        list.getData(101);
        list.getData(102);

        list.updateRate(101,25);
        list.updateQuantity(101,57);
        list.getData(101);

        list.updateRate(102,245);
        list.updateQuantity(102,43);
        list.getData(101);
}
```

```
Item Code: 101
Item Name: biscuit
Item Rate: 20
Item Quantity: 50
Item Code: 102
Item Name: tea
Item Rate: 234
Item Quantity: 38
Item Code: 101
Item Name: biscuit
Item Rate: 25
Item Quantity: 107
Item Code: 101
Item Name: biscuit
Item Rate: 25
Item Quantity: 107
```

```
/*
    ASSIGNMENT 3              * QUESTION 13

Design a BALANCE class with account number, balance and date of last update.
Consider a TRANSACTION class with account number, date of transaction, amount
and transaction type (W for withdrawal and D for deposit). If it is a withdrawal
check whether the amount is available or not. Transaction object will make
necessary updates in the balance class.

    NAME: Anirudh Modi    ROLL:00231050131    DATE:03/10/2024
*/

#include<iostream>
#include<string>
#include<ctime>
using namespace std;

class balance
{
        string acc_num;
        double bal;
        tm last_time;

        public:
                balance(const string &s, double balance)
                {
                        this->acc_num=s;
                        this->bal=balance;
                        time_t now = time(0);
                        last_time = *localtime(&now);
                }

                string getAccount()
                {
                        return this->acc_num;
                }

                void display()
                {
                        cout<<"Account Number : "<<acc_num<<endl;
                        cout<<"Account Balance : "<<bal<<endl;
                        cout<<"Last Update time : "<<asctime(&last_time)<<endl;
                }

                void update(double x, char type)
                {
                        if(type=='D')
```

```
                {
                        this->bal+=x;
                }
                else if(type=='W')
                {
                        if(x>this->bal)
                                cout<<"Insufficient balance to withdraw!! "<<endl;
                        else
                                this->bal-=x;
                }
                time_t now = time(0);
                last_time = *localtime(&now);
        }
};


class transaction
{
        string acc_num;
        double amount;
        char transaction_type;
        tm last;

        public:
                transaction(const string &acc_num, double x, char type)
                {
                        this->acc_num=acc_num;
                        this->amount=x;
                        this->transaction_type = type;
                        time_t now = time(0);
                        last = *localtime(&now);
                }

                void Transaction(balance &saving_acc)
                {
                        saving_acc.update(this->amount, this->transaction_type);
                        time_t now = time(0);
                        last = *localtime(&now);
                }

                void display()
                {
                        cout<<"Account number : "<<this->acc_num<<endl;
                        cout<<"Transaction amount : "<<this->amount<<endl;
                        cout<<"Transaction type : "<<this->transaction_type<<endl;
                        cout<<"Last update time : "<<asctime(&(this->last))<<endl;
                }
```

```
};


int main()
{
        balance savings_acc("SBI002210501021",50.66);
        savings_acc.display();
        transaction transaction_(savings_acc.getAccount(),319,'W');
        transaction_.Transaction(savings_acc);
        transaction_.display();
        savings_acc.display();

return 0;
}
```

```
Account Number : SBI002210501021
Account Balance : 50.66
Last Update time : Wed Nov 13 14:57:53 2024

Insufficient balance to withdraw!!
Account number : SBI002210501021
Transaction amount : 319
Transaction type : W
Last update time : Wed Nov 13 14:57:53 2024

Account Number : SBI002210501021
Account Balance : 50.66
Last Update time : Wed Nov 13 14:57:53 2024
```