Salman Ahmad

04072113050

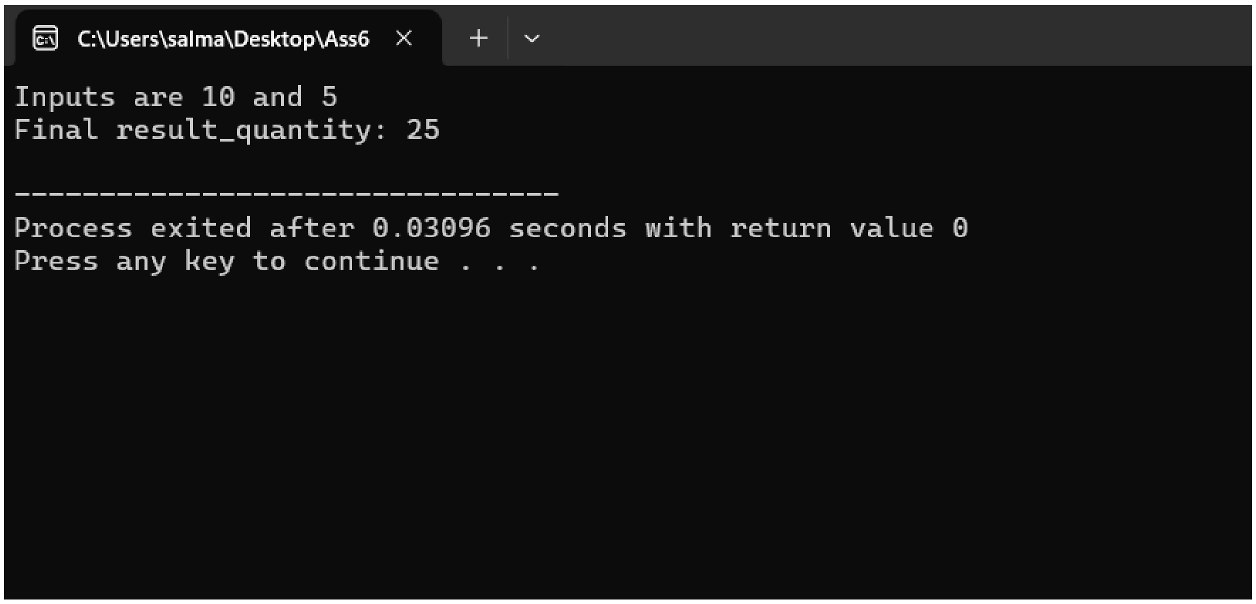
BSCS 6th Sem

CS-121 OOP Assignment 6

Q1.

```
#include <iostream>

class Number {
private:
   int quantity;

public:
   // Constructor
   Number(int quantity) : quantity(quantity) {}

   // Overloading >= operator to compare quantities
   bool operator>=(const Number& other) const {
      return quantity >= other.quantity;
   }

   // Overloading + operator to add quantities
   Number operator+(const Number& other) const {
      return Number(quantity + other.quantity);
   }
```

```
  // Overloading - operator to subtract quantities
  Number operator-(const Number& other) const {
    return Number(quantity - other.quantity);
  }

  // Overload == operator to check equality of quantities
  bool operator==(const Number& other) const {
    return quantity == other.quantity;
  }

  // Getter function
  int getQuantity() const {
    return quantity;
  }
};

using namespace std;

int main() {

  Number item1(10);
  Number item2(5);

  cout<<"Inputs are 10 and 5\n";
  if (item1 >= item2) {
    Number result_quantity = item1 + item2 + item2 + item2;
    cout << "Final result_quantity: " << result_quantity.getQuantity() << endl;
  } else {
        Number result_quantity = item1 - item2 - item2 - item2;
        cout << "Final result_quantity: " << result_quantity.getQuantity() << endl;
    }



    if (item1 == item2) {
        cout << "They are equal" << endl;
    }

    return 0;
}
```



C:\Users\salma\Desktop\Ass6

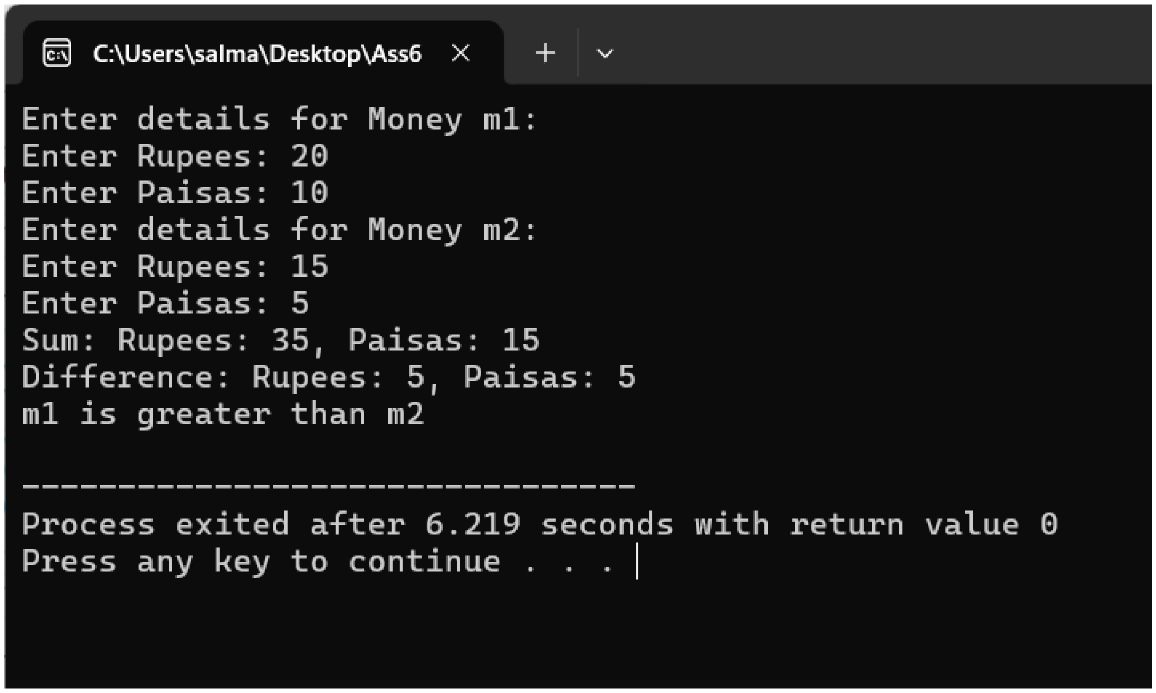
```
Inputs are 10 and 5
Final result_quantity: 25

--------------------------------
Process exited after 0.03096 seconds with return value 0
Press any key to continue . . .
```

Q2.

```
#include <iostream>

using namespace std;

class Money {
private:
    int Rupees;
    int Paisas;

public:
    // Parameterized constructor with default values
    Money(int rupees = 0, int paisas = 0) : Rupees(rupees), Paisas(paisas) {}

    // Overloading + operator to add two Money objects
    Money operator+(const Money& other) const {
        Money result;
        result.Rupees = Rupees + other.Rupees;
        result.Paisas = Paisas + other.Paisas;

        // Adjusting Paisas if it exceeds 100
        if (result.Paisas >= 100) {
            result.Rupees += result.Paisas / 100;
            result.Paisas %= 100;
        }

        return result;
    }
```

```
// Overloading - operator to subtract two Money objects
Money operator-(const Money& other) const {
  Money result;
  result.Rupees = Rupees - other.Rupees;
  result.Paisas = Paisas - other.Paisas;

  // Adjusting Paisas if it goes negative
  if (result.Paisas < 0) {
    result.Rupees -= 1;
    result.Paisas += 100;
  }

  return result;
}

// Overloading << operator for output
friend ostream& operator<<(ostream& os, const Money& money) {
  os << "Rupees: " << money.Rupees << ", Paisas: " << money.Paisas;
  return os;
}

// Overloading >> operator for input
friend istream& operator>>(istream& is, Money& money) {
  cout << "Enter Rupees: ";
  is >> money.Rupees;
  cout << "Enter Paisas: ";
  is >> money.Paisas;
  return is;
```

```
    }

    // Compare function
    int compare(const Money& other) const {
        if (Rupees < other.Rupees) {
            return -1;
        } else if (Rupees > other.Rupees) {
            return 1;
        } else {
            if (Paisas < other.Paisas) {
                return -1;
            } else if (Paisas > other.Paisas) {
                return 1;
            } else {
                return 0;
            }
        }
    }
};

int main() {
    Money m1, m2;

    // Input for m1 and m2
    cout << "Enter details for Money m1:" << endl;
    cin >> m1;

    cout << "Enter details for Money m2:" << endl;
    cin >> m2;
    // Addition
    Money sum = m1 + m2;
    cout << "Sum: " << sum << endl;

    // Subtraction
    Money diff = m1 - m2;
    cout << "Difference: " << diff << endl;

    // Comparison
    int result = m1.compare(m2);
    if (result < 0) {
        cout << "m1 is less than m2" << endl;
    } else if (result > 0) {
        cout << "m1 is greater than m2" << endl;
    } else {
        cout << "m1 is equal to m2" << endl;
    }

    return 0;
}
```



```
Enter details for Money m1:
Enter Rupees: 20
Enter Paisas: 10
Enter details for Money m2:
Enter Rupees: 15
Enter Paisas: 5
Sum: Rupees: 35, Paisas: 15
Difference: Rupees: 5, Paisas: 5
m1 is greater than m2

--------------------------------
Process exited after 6.219 seconds with return value 0
Press any key to continue . . .
```

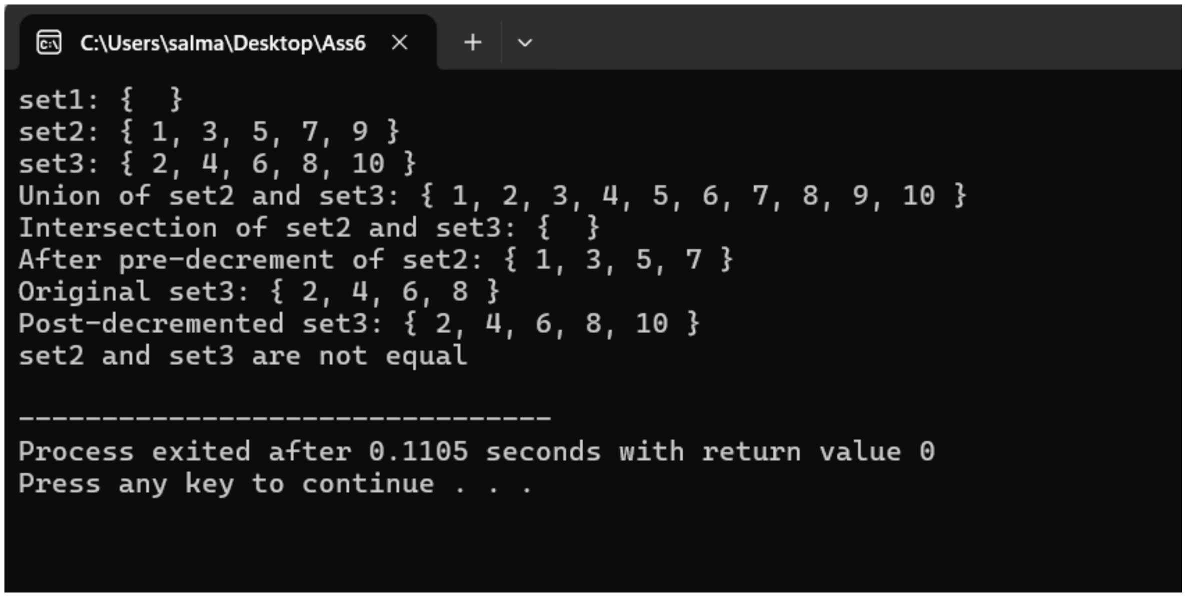
Q3.

```
#include <iostream>

using namespace std;



const int MAX_ELEMENTS = 101; // Represents integers 0 to 100


class IntegerSet {

private:

  bool elements[MAX_ELEMENTS]; // Array to represent set elements


public:

  // Default constructor initializes an empty set

  IntegerSet() {

    for (int i = 0; i < MAX_ELEMENTS; ++i) {
        elements[i] = false;
    }
}

// Constructor with array initialization
IntegerSet(const int arr[], int size) {
    for (int i = 0; i < MAX_ELEMENTS; ++i) {
        elements[i] = false;
    }

    // Set elements to true if they exist in the array and are in range 0-100
    for (int i = 0; i < size; ++i) {
        if (arr[i] >= 0 && arr[i] <= 100) {
            elements[arr[i]] = true;
        }
    }
}

// Union operator (+)
IntegerSet operator+(const IntegerSet& other) const {
    IntegerSet result;
    for (int i = 0; i < MAX_ELEMENTS; ++i) {
        result.elements[i] = (elements[i] || other.elements[i]);
    }
    return result;
}

// Intersection operator (*)
IntegerSet operator*(const IntegerSet& other) const {
    IntegerSet result;
    for (int i = 0; i < MAX_ELEMENTS; ++i) {
        result.elements[i] = (elements[i] && other.elements[i]);
    }
    return result;
}

// Pre-decrement operator (--set)
IntegerSet& operator--() {
    for (int i = MAX_ELEMENTS - 1; i >= 0; --i) {
        if (elements[i]) {
            elements[i] = false;
            break;
        }
    }
    return *this;
}

// Post-decrement operator (set--)
IntegerSet operator--(int) {
    IntegerSet temp(*this); // Create a copy of the current object
    --(*this); // Use the pre-decrement operator
    return temp; // Return the copy of the original object
}

// Output operator (<<)
friend std::ostream& operator<<(std::ostream& os, const IntegerSet& set) {
    os << "{ ";
    bool first = true;
```

```
        for (int i = 0; i < MAX_ELEMENTS; ++i) {
            if (set.elements[i]) {
                if (!first) {
                    os << ", ";
                }
                os << i;
                first = false;
            }
        }
        os << " }";
        return os;
    }

    // Equality operator (==)
    bool operator==(const IntegerSet& other) const {
        for (int i = 0; i < MAX_ELEMENTS; ++i) {
            if (elements[i] != other.elements[i]) {
                return false;
            }
        }
        return true;
    }
};

int main() {


    // Test cases
    IntegerSet set1; // Empty set
```

```
int arr2[] = { 1, 3, 5, 7, 9 };
IntegerSet set2(arr2, sizeof(arr2) / sizeof(arr2[0])); // Initializing set with array
int arr3[] = { 2, 4, 6, 8, 10 };
IntegerSet set3(arr3, sizeof(arr3) / sizeof(arr3[0])); // Initializing another set with array

cout << "set1: " << set1 << endl;
cout << "set2: " << set2 << endl;
cout << "set3: " << set3 << endl;

// Test union operator (+)
IntegerSet unionSet = set2 + set3;
cout << "Union of set2 and set3: " << unionSet << endl;

// Testing intersection operator (*)
IntegerSet intersectSet = set2 * set3;
cout << "Intersection of set2 and set3: " << intersectSet << endl;

// Testing pre-decrement operator (--set)
--set2;
cout << "After pre-decrement of set2: " << set2 << endl;

// Testing post-decrement operator (set--)
IntegerSet postDecSet = set3--;
cout << "Original set3: " << set3 << endl;
cout << "Post-decremented set3: " << postDecSet << endl;

// Testing equality operator (==)
if (set2 == set3) {
  cout << "set2 and set3 are equal" << endl;
```

```
    }
    else {
        cout << "set2 and set3 are not equal" << endl;
    }

    return 0;
}
```



C:\Users\salma\Desktop\Ass6

```
set1: {  }
set2: { 1, 3, 5, 7, 9 }
set3: { 2, 4, 6, 8, 10 }
Union of set2 and set3: { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
Intersection of set2 and set3: {  }
After pre-decrement of set2: { 1, 3, 5, 7 }
Original set3: { 2, 4, 6, 8 }
Post-decremented set3: { 2, 4, 6, 8, 10 }
set2 and set3 are not equal

--------------------------------
Process exited after 0.1105 seconds with return value 0
Press any key to continue . . .
```