```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#define UNIVERSAL_SIZE 26

void createBitVector(char set[], int bit_vector[]) {
    memset(bit_vector, 0, UNIVERSAL_SIZE * sizeof(int));
    for (int i = 0; set[i] != '\0'; i++) {
        if (isalpha(set[i])) {
            bit_vector[tolower(set[i]) - 'a'] = 1;
        }
    }
}

void printBitVector(int bit_vector[]) {
    for (int i = 0; i < UNIVERSAL_SIZE; i++) {
        printf("%d", bit_vector[i]);
    }
    printf("\n");
}
```

```
void performUnion(int set1[], int set2[], int result[]) {
    for (int i = 0; i < UNIVERSAL_SIZE; i++) {
        result[i] = set1[i] | set2[i];
    }
}
void performIntersection(int set1[], int set2[], int result[]) {
    for (int i = 0; i < UNIVERSAL_SIZE; i++) {
        result[i] = set1[i] & set2[i];
    }
}
void performComplement(int set[], int result[]) {
    for (int i = 0; i < UNIVERSAL_SIZE; i++) {
        result[i] = !set[i];
    }
}
void performDifference(int set1[], int set2[], int result[]) {
    for (int i = 0; i < UNIVERSAL_SIZE; i++) {
        result[i] = set1[i] & !set2[i];
    }
}
int main() {
    char set1[UNIVERSAL_SIZE + 1], set2[UNIVERSAL_SIZE + 1];
    int set1_bit[UNIVERSAL_SIZE], set2_bit[UNIVERSAL_SIZE], result[UNIVERSAL_SIZE];
    int choice;
    printf("Enter elements for Set 1 (lowercase letters only): ");
    scanf("%s", set1);
    printf("Enter elements for Set 2 (lowercase letters only): ");
    scanf("%s", set2);
```

```
createBitVector(set1, set1_bit);
createBitVector(set2, set2_bit);
printf("\nBit Vector of Set 1:\n");
printBitVector(set1_bit);
printf("Bit Vector of Set 2:\n");
printBitVector(set2_bit);
do {
   printf("\nOperations:\n");
   printf("1. Union\n");
   printf("2. Intersection\n");
   printf("3. Complement\n");
   printf("4. Difference (Set1 - Set2)\n");
   printf("5. Exit\n");
   printf("Choose an option: ");
   scanf("%d", &choice);

   switch (choice) {
      case 1:
         performUnion(set1_bit, set2_bit, result);
         printf("\nUnion:\n");
         printBitVector(result);
         break;
      case 2:
         performIntersection(set1_bit, set2_bit, result);
         printf("\nIntersection:\n");
         printBitVector(result);
         break;
```

```
            case 3:
                performComplement(set1_bit, result);
                printf("\nComplement of Set 1:\n");
                printBitVector(result);
                performComplement(set2_bit, result);
                printf("\nComplement of Set 2:\n");
                printBitVector(result);
                break;
            case 4:
                performDifference(set1_bit, set2_bit, result);
                printf("\nDifference (Set1 - Set2):\n");
                printBitVector(result);
                performDifference(set2_bit, set1_bit, result);
                printf("\nDifference (Set2 - Set1):\n");
                printBitVector(result);
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice! Try again.\n");
        }
    } while (choice != 5);
    return 0;
}
```