```
#include <stdio.h>
#include <stdlib.h>

typedef struct ArrayADT {
   int *data;
   int size;
   int capacity;
} ArrayADT;


ArrayADT *createArray(int capacity) {
   ArrayADT *array = (ArrayADT *)malloc(sizeof(ArrayADT));
   array->data = (int *)malloc(capacity * sizeof(int));
   array->size = 0;
   array->capacity = capacity;
   return array;
}

void init(ArrayADT *array) {
   for (int i = 0; i < array->capacity; i++) {
      array->data[i] = rand() % 100; // Populate with random values
   }
   array->size = array->capacity;
}

void append(ArrayADT *array, int value) {
   if (array->size < array->capacity) {
      array->data[array->size++] = value;
   } else {
      printf("Array is full. Cannot append.\n");
   }
}
```

```
void insertAtIndex(ArrayADT *array, int value, int index) {
    if (index >= 0 && index < array->size) {
        for (int i = array->size; i > index; i--) {
            array->data[i] = array->data[i - 1];
        }
        array->data[index] = value;
        array->size++;
    } else {
        printf("Index out of bounds.\n");
    }
}

void removeAtIndex(ArrayADT *array, int index) {
    if (index >= 0 && index < array->size) {
        for (int i = index; i < array->size - 1; i++) {
            array->data[i] = array->data[i + 1];
        }
        array->size--;
    } else {
        printf("Index out of bounds.\n");
    }
}

void display(const ArrayADT *array) {
    for (int i = 0; i < array->size; i++) {
        printf("%d ", array->data[i]);
    }
    printf("\n");
}

int Max(const ArrayADT *array) {
```

```
    int max = array->data[0];
    for (int i = 1; i < array->size; i++) {
        if (array->data[i] > max) {
            max = array->data[i];
        }
    }
    return max;
}

int Min(const ArrayADT *array) {
    int min = array->data[0];
    for (int i = 1; i < array->size; i++) {
        if (array->data[i] < min) {
            min = array->data[i];
        }
    }
    return min;
}

void reverse(ArrayADT *array) {
    for (int i = 0; i < array->size / 2; i++) {
        int temp = array->data[i];
        array->data[i] = array->data[array->size - i - 1];
        array->data[array->size - i - 1] = temp;
    }
}

void merge(ArrayADT *array1, const ArrayADT *array2) {
    for (int i = 0; i < array2->size && array1->size < array1->capacity; i++) {
        array1->data[array1->size++] = array2->data[i];
    }
```

```
}

void freeA(ArrayADT *array) {
   free(array->data);
   free(array);
}

#include "header.h"
#include "logic.c"
int main() {
   srand(42);
   ArrayADT *array1 = createArray(100);
   ArrayADT *array2 = createArray(100);
   int choice, value, index;

   do {
      printf("\nMenu:\n");
      printf("1. Initialize Array 1\n");
      printf("2. Append Element to Array 1\n");
      printf("3. Insert Element in Array 1 at Index\n");
      printf("4. Remove Element from Array 1 at Index\n");
      printf("5. Display Array 1\n");
      printf("6. Max Element in Array 1\n");
      printf("7. Min Element in Array 1\n");
      printf("8. Reverse Array 1\n");
      printf("9. Initialize Array 2 and Merge with Array 1\n");
      printf("0. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
```

```
        case 1:
            init(array1);
            break;
        case 2:
            printf("Enter value to append: ");
            scanf("%d", &value);
            append(array1, value);
            break;
        case 3:
            printf("Enter value to insert: ");
            scanf("%d", &value);
            printf("Enter index: ");
            scanf("%d", &index);
            insertAtIndex(array1, value, index);
            break;
        case 4:
            printf("Enter index to remove: ");
            scanf("%d", &index);
            removeAtIndex(array1, index);
            break;
        case 5:
            display(array1);
            break;
        case 6:
            printf("Max Element: %d\n", Max(array1));
            break;
        case 7:
            printf("Min Element: %d\n", Min(array1));
            break;
        case 8:
            reverse(array1);
```

```
            break;
        case 9:
            init(array2); // Initialize second array
            merge(array1, array2);
            break;
        case 0:
            freeA(array1);
            freeA(array2);
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  } while (choice != 0);

  return 0;
}
```

```
Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 1

Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 2
Enter value to append: 6
Array is full. Cannot append.
```

```
Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 4
Enter index to remove: 7

Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 2
Enter value to append: 5
```

```
Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 5
75 0 69 56 83 79 16 9 69 9 50 71 99 36 25 88 12 51 76 7 30 86 7 90 40 36 91 45 15 82 97 88 45 6 47 49 72 67 55 69 23 43 68 60 99
81 49 35 36 32 23 9 70 87 2 45 30 67 69 76 35 38 71 2 86 65 5 21 97 92 76 25 67 97 45 9 13 42 79 54 23 63 64 62 58 91 40 92 33 46
 67 42 17 27 14 93 66 82 48 5

Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 6
Max Element: 99
```

```
Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 7
Min Element: 0

Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 8
```

```
Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 9

Menu:
1. Initialize Array 1
2. Append Element to Array 1
3. Insert Element in Array 1 at Index
4. Remove Element from Array 1 at Index
5. Display Array 1
6. Max Element in Array 1
7. Min Element in Array 1
8. Reverse Array 1
9. Initialize Array 2 and Merge with Array 1
0. Exit
Enter your choice: 5
5 48 82 66 93 14 27 17 42 67 46 33 92 40 91 58 62 64 63 23 54 79 42 13 9 45 97 67 25 76 92 97 21 5 65 86 2 71 38 35 76 69 67 30 4
5 2 87 70 9 23 32 36 35 49 81 99 60 68 43 23 69 55 67 72 49 47 6 45 88 97 82 15 45 91 36 40 90 7 86 30 7 76 51 12 88 25 36 99 71
50 9 69 9 16 79 83 56 69 0 75
```