Task 1: Shift Left k Cells

```
#include <stdio.h>
void shiftLeft(int source[], int k, int size) {
for (int i = 0; i < size - k; i++) {
 source[i] = source[i + k];
 }
 for (int i = size - k; i < size; i++) {
 source[i] = 0;
}
}
int main() {
int source[] = {10, 20, 30, 40, 50, 60};
int k = 3;
 int size = sizeof(source) / sizeof(source[0]);
printf("Before shifting: [ ");
for (int i = 0; i < size; i++) {
 printf("%d ", source[i]);
 }
 printf("]\n");
 shiftLeft(source, k, size);
  printf("After shifting: [ ");
 for (int i = 0; i < size; i++) {
 printf("%d ", source[i]);
 }
 printf("]\n");
return 0;
}
```

Task 2: Shift Right k Cells

```
#include <stdio.h>
void shiftRight(int source[], int k, int size) {
 for (int i = size - 1; i >= k; i--) {
 source[i] = source[i - k];
 }
 for (int i = 0; i < k; i++) {
 source[i] = 0;
 }
 }
int main() {
 int source[] = {10, 20, 30, 40, 50, 60};
 int k = 3; // Number of cells to shift
 int size = sizeof(source) / sizeof(source[0]); // Calculate the size of the array
 printf("Original array: [ ");
 for (int i = 0; i < size; i++) {
 printf("%d ", source[i]);
 }
 printf("]\n");
shiftRight(source, k, size);
printf("After shifting: [ ");
 for (int i = 0; i < size; i++) {
 printf("%d ", source[i]);
 }
printf("]\n");
return 0;
}
```

Task 3: Rotate Left k cells

```
#include <stdio.h>
void rotateLeft(int source[], int k, int size) {
int temp[k];
for (int i = 0; i < k; i++) {
temp[i] = source[i];
}
for (int i = k; i < size; i++) {
source[i - k] = source[i];
}
for (int i = size - k; i < size; i++) {
source[i] = temp[i - (size - k)];
}
}
int main() {
int source[] = {10, 20, 30, 40, 50, 60};
int k = 3;
int size = sizeof(source) / sizeof(source[0]);
printf("Original array: [ ");
for (int i = 0; i < size; i++) {
printf("%d ", source[i]);
}
printf("]\n");
rotateLeft(source, k, size);
printf("After rotating: [ ");
for (int i = 0; i < size; i++) {
printf("%d ", source[i]);
}
printf("]\n");
return 0;
}
```

Task 4: Rotate Right k cells

```
#include <stdio.h>
void rotateRight(int source[], int k, int size) {
int temp[k];
for (int i = size - k; i < size; i++) {
temp[i - (size - k)] = source[i];
}
for (int i = size - k - 1; i >= 0; i--) {
source[i + k] = source[i];
}
for (int i = 0; i < k; i++) {
source[i] = temp[i];
}
}
int main() {
int source[] = {10, 20, 30, 40, 50, 60};
int k = 3;
int size = sizeof(source) / sizeof(source[0]);
printf("Original array: [ ");
for (int i = 0; i < size; i++) {
printf("%d ", source[i]);
}
printf("]\n");
rotateRight(source, k, size);
printf("After rotating: [ ");
for (int i = 0; i < size; i++) {
printf("%d ", source[i]);
}
printf("]\n");
return 0;
}
```

Task 5: Remove an element from an array

```
#include <stdio.h>
void removeElement(int source[], int size, int idx) {
if (idx < 0 || idx >= size) {
printf("Invalid index. Element not removed.\n");
return;
}
for (int i = idx; i < size - 1; i++) {
source[i] = source[i + 1];
}
source[size - 1] = 0;
}
int main() {
int source[] = {10, 20, 30, 40, 50, 0, 0};
int size = 7;
int idx = 2;
printf("Original array: [ ");
for (int i = 0; i < size; i++) {
printf("%d ", source[i]);
}
printf("]\n");
removeElement(source, size, idx);
printf("Array after removal: [ ");
for (int i = 0; i < size; i++) {
printf("%d ", source[i]);
}
printf("]\n");
return 0;
}
```

Task 6: Remove all occurrences of a particular element from an array

```
#include <stdio.h>
void removeAll(int source[], int size, int element) {
int new_size = 0;
for (int i = 0; i < size; i++) {
if (source[i] != element) {
source[new_size] = source[i];
new_size++;
}
}
for (int i = new_size; i < size; i++) {
source[i] = 0;
}
}
int main() {
int source[] = {10, 2, 30, 2, 50, 2, 2, 0, 0};
int size = 9;
int element = 2;
removeAll(source, size, element);
for (int i = 0; i < size; i++) {
printf("%d ", source[i]);
}
return 0;
}
```

Task 7: Repetition

```
#include <stdio.h>
#include <stdbool.h>
bool hasDuplicateRepetitions(int arr[], int size)
{
int i, j;
bool foundDuplicate = false;
for (i = 0; i < size; i++) {
int count = 0;
for (j = 0; j < size; j++) {
if (arr[i] == arr[j]) {
count++;
foundDuplicate = true;
}
}
return foundDuplicate;
}
int main() {
int input1[] = {4, 5, 6, 6, 4, 3, 6, 4};
int input2[] = {3, 4, 6, 3, 4, 7, 4, 6, 8, 6, 6};
bool result1 = hasDuplicateRepetitions(input1, sizeof(input1) / sizeof(input1[0]));
bool result2 = hasDuplicateRepetitions(input2, sizeof(input2) / sizeof(input2[0]));
 if (result1) {
printf("Output for input1: True\n");
} else {
printf("Output for input1: False\n");
}
if (result2) {
```

```
printf("Output for input2: True\n");

} else {

printf("Output for input2: False\n");

}

return 0;

}
```