1. main.c

```
#include "array.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
   srand(time(0));
   array a, b, *c = NULL;
   int choice, index;
   do {
      printf("\nMenu:\n");
      printf("1. init()\n");
      printf("2. append()\n");
      printf("3. insert_at_index()\n");
      printf("4. remove_at_index()\n");
      printf("5. display()\n");
      printf("6. max()\n");
      printf("7. min()\n");
      printf("8. reverse()\n");
      printf("9. merge()\n");
      printf("0. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch(choice){
         case 1:
            init(&a, 10);
            break;
         case 2:
            append(&a, rand() % 100);
```

```
    break;
case 3:
    printf("Enter index of element to insert: ");
    scanf("%d", &index);
    insert_at_index(&a , index, rand() % 100);
    break;
case 4:
    printf("Enter index to remove: ");
    scanf("%d", &index);
    remove_at_index(&a, index);
    break;
case 5:
    display(&a);
    break;
case 6:
    printf("Maximum element: %d\n", max(a));
    break;
case 7:
    printf("Minimum element: %d\n", min(a));
    break;
case 8:
    reverse(&a);
    break;
case 9:
    init(&b, 10);
    for(int i = 0; i < b.size; i++){
        b.A[i] = rand() % 100;
    }
    b.length = b.size;
    printf("First Array : ");
    display(&a);
```

```
            printf("\nSecond Array : ");
            display(&b);
            printf("\nMerged Array : ");
            c = merge(&a, &b);
            display(c);
            break;
        case 0:
            printf("Exiting program.\n");
            break;
        default:
            printf("Invalid choice! Please try again.\n");
      }
   } while (choice != 0);
   return 0;
}
```

2. array.h

```
typedef struct array{
   int *A;
   int size;
   int length;
}array;


void init(array *a, int size);
void append(array *a, int element);
void insert_at_index(array *a, int position, int element);
void remove_at_index(array *a, int position);
void display(array *a);
int max(array a);
int min(array a);
void reverse(array *a);
array *merge(array *a, array *b);
```

3. array.c

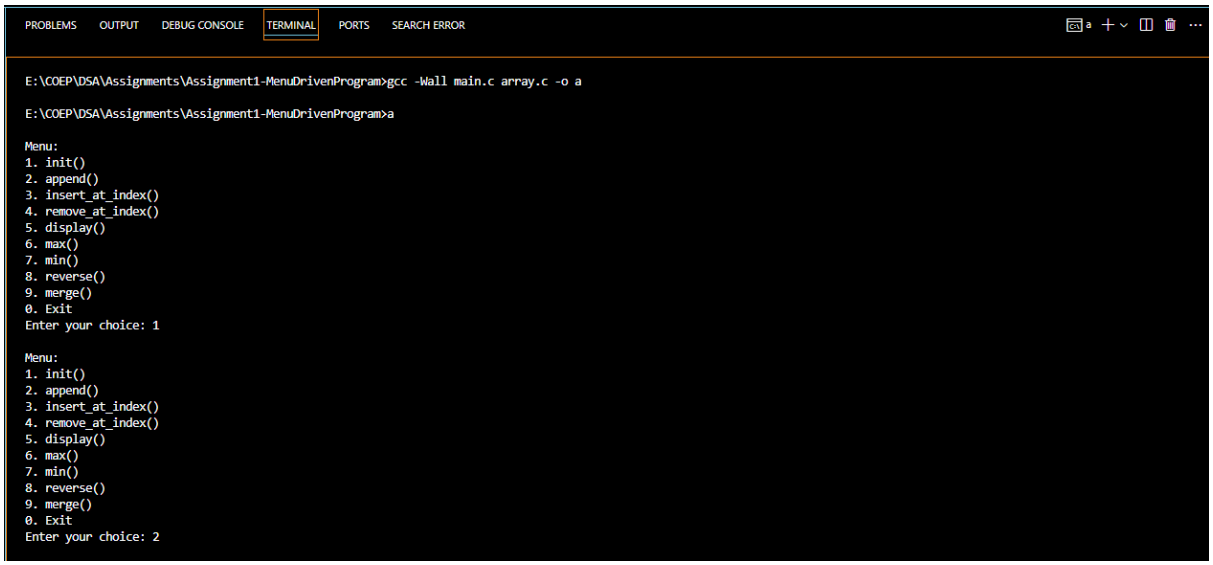
```
#include "array.h"
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

void init(array *a, int size){
   a -> A = (int *)malloc(sizeof(int) * size);
   a -> size = size;
   a -> length = 0;
   return;
}

void append(array *a, int element){
   if(a -> length < a -> size){
      a -> A[a -> length++] = element;
   }
   return;
}

void insert_at_index(array *a, int position, int element){
   if(position >= 0 && position <= (a -> length) && a -> length < a -> size){
      for(int i = a -> length; i > position; i--){
         a -> A[i] = a -> A[i - 1];
      }
      a -> A[position] = element;
      a -> length++;
   }
   else if(a -> length == a -> size){
      printf("Array is full\n");
   }
```

```
    else{
        printf("Invalid Position\n");
    }
}

void remove_at_index(array *a, int position){
    if(position >= 0 && position < a -> length){
        for(int i = position; i < a -> length - 1; i++){
            a -> A[i] = a -> A[i + 1];
        }
        a -> length--;
    }
    else{
        printf("Invalid Position\n");
    }
}

void display(array *a){
    for(int i = 0; i < a -> length; i++){
        printf("%d |", a -> A[i]);
    }
    return;
}

int max(array a){
    int max = INT_MIN;
    for(int i = 0; i < a.length; i++){
        if(a.A[i] > max){
            max = a.A[i];
        }
    }
```

```
    return max;
}

int min(array a){
    int min = INT_MAX;
    for(int i = 0; i < a.length; i++){
        if(a.A[i] < min){
            min = a.A[i];
        }
    }
    return min;
}

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

array *merge(array *a, array *b){
    array *c = (array*)malloc(sizeof(array));
    init(c, a -> length + b -> length);
    for(int i = 0; i < a -> length; i++){
        c -> A[i] = a -> A[i];
    }
    for(int i = 0; i < b -> length; i++){
        c -> A[a -> length + i] = b -> A[i];
    }
    c -> length = a -> length + b -> length;
```

```
    return c;
}
```

1. init() function and append() function



2. append() function and display() function

3. insert_at_index() function and display() function

```
Menu:
1. init()
2. append()
3. insert_at_index()
4. remove_at_index()
5. display()
6. max()
7. min()
8. reverse()
9. merge()
0. Exit
Enter your choice: 3
Enter index of element to insert: 1

Menu:
1. init()
2. append()
3. insert_at_index()
4. remove_at_index()
5. display()
6. max()
7. min()
8. reverse()
9. merge()
0. Exit
Enter your choice: 5
95 |20 |
```

4. append() function and display() function

```
Menu:
1. init()
2. append()
3. insert_at_index()
4. remove_at_index()
5. display()
6. max()
7. min()
8. reverse()
9. merge()
0. Exit
Enter your choice: 2

Menu:
1. init()
2. append()
3. insert_at_index()
4. remove_at_index()
5. display()
6. max()
7. min()
8. reverse()
9. merge()
0. Exit
Enter your choice: 5
95 |20 |35 |
```

5. remove_at_index() function and display() function

6. max() function and min() func

7. reverse() function and display() function

8. merge() function and exit choice