

## Data Structure Lab Report

| Only for course Teacher               |           |                   |            |            |               |            |
|---------------------------------------|-----------|-------------------|------------|------------|---------------|------------|
|                                       |           | Needs Improvement | Developing | Sufficient | Above Average | Total Mark |
| <b>Allocate mark &amp; Percentage</b> |           | <b>25%</b>        | <b>50%</b> | <b>75%</b> | <b>100%</b>   | <b>25</b>  |
| <b>Understanding</b>                  | <b>3</b>  |                   |            |            |               |            |
| <b>Analysis</b>                       | <b>4</b>  |                   |            |            |               |            |
| <b>Implementation</b>                 | <b>8</b>  |                   |            |            |               |            |
| <b>Report Writing</b>                 | <b>10</b> |                   |            |            |               |            |
| <b>Total obtained mark</b>            |           |                   |            |            |               |            |
| <b>Comments</b>                       |           |                   |            |            |               |            |

**Semester: Fall 2024**

**Student Name: Md. Jakaria Nur**

**Student ID: 232-35-252**

**Batch: SWE-41**

**Section: A2**

**Course Code: SE 132**

**Course Name: Data Structure Lab**

**Course Teacher Name: Md. Abdul Hye Zebon**

**Designation: Lecturer**

**Submission Date: 12/12/24**

Date: 14-09-2024

Problem No: 01

Problem Statement: How to replace a specific element in an array.

```
#include<stdio.h>

int main(){
    int arr[]={10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int len = sizeof(arr)/sizeof(arr[0]);
    printf("Array element are: \n");
    for(int i = 0; i<len ; i++){
        printf("%d ", arr[i]);
    }
    int value, position;
    printf("\nEnter the position do you want to replacement : ");
    scanf("%d", &position);
    printf("Enter element for the replacement : ");
    scanf("%d", &value);
    arr[position-1] = value ;
    for(int i = 0; i<len ; i++){
        printf("%d ", arr[i]);
    }
}
```

| main.c  | Output  |
|---|---|
| <pre>1 #include&lt;stdio.h&gt; 2- int main(){ 3     int arr[]={10,20,30,40,50,60,70,80,90,100}; 4     int len = sizeof(arr)/sizeof(arr[0]); 5     printf("Array element are: \n"); 6-     for(int i = 0; i&lt;len ; i++){ 7         printf("%d ",arr[i]); 8     } 9 10    int value,position; 11    printf("\nEnter the position do you want to replacement : "); 12    scanf("%d",&amp;position); 13    printf("Enter element for the replacement : "); 14    scanf("%d",&amp;value); 15    arr[position-1] = value ; 16-    for(int i = 0; i&lt;len ; i++){</pre> | <pre>Array element are: 10 20 30 40 50 60 70 80 90 100 Enter the position do you want to replacement : 3 Enter element for the replacement : 5555 10 20 5555 40 50 60 70 80 90 100  === Code Execution Successful ===</pre> |

Date: **14-09-2024**

Problem No: **02**

Problem Statement: **Delete Specific Element in the Array.**

```
#include <stdio.h>

int main() {
    int arr[100], i, size, position;
    printf("Enter Array Size: ");
    scanf("%d", &size);
    printf("Enter %d Array Element: ", size);
    for(i=0; i<size; i++){
        scanf("%d", &arr[i]);
    }
    printf("The Array Element Are: \n");
    for(i=0; i<size; i++){
        printf("%d ", arr[i]);
    }
    printf("\nEnter the position you want to delete: ");
    scanf("%d", &position);
    for(i=position-1; i<size-1; i++){
        arr[i]=arr[i+1];
    }
    arr[size-1]=0;
    printf("The Array Element After Deletion: \n");
    for(i=0; i<size-1; i++){
        printf("%d ", arr[i]);
    }
    return 0;
}
```

| main.c  | Run | Output   |
|---|-----|--|
| <pre>7-  for(i=0;i&lt;size;i++){ 8      scanf("%d", &amp;arr[i]); 9  } 10 11  printf("The Array Element Are: \n"); 12- for(i=0;i&lt;size;i++){ 13      printf("%d ", arr[i]); 14  } 15 16  printf("\nEnter the position you want to delete: "); 17  scanf("%d", &amp;position); 18- for(i=position-1;i&lt;size-1;i++){ 19      arr[i]=arr[i+1]; 20  } 21      arr[size-1]=0; 22  printf("The Array Element Atfer Deletion: \n"); 23- for(i=0;i&lt;size-1;i++){ 24      printf("%d ", arr[i]); 25  }</pre> |     | <pre>Enter Array Size: 5 Enter 5 Array Element: 1 2 3 4 5 The Array Element Are: 1 2 3 4 5 Enter the position you want to delete: 3 The Array Element Atfer Deletion: 1 2 4 5  === Code Execution Successful ===</pre> |

Date: 14-09-2024

Problem No: 03

Problem Statement: Find the Maximum Number in the Array.

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

```
int main(){
```

```
int i, size;
```

```
printf("Enter any array size: ");
```

```
scanf("%d", &size);
```

```
int arr[size];
```

```
printf("Enter %d numbers: ", size);
```

```
for(i=0; i<size; i++){
```

```
scanf("%d", &arr[i]);
```

```
}
```

```
printf("The array element are: \n");
```

```
for(i=0;i<size;i++){
```

```
    printf("%d ", arr[i]);
```

```
}
```

```
printf("\nThe maximum number is: ");
```

```

int maximum = arr[0];
for(i=1; i<size; i++){
    if(arr[i] > maximum)
    {
        maximum=arr[i];
    }
}
printf("%d", maximum);
return 0;
}

```

| main.c  | Run | Output   |
|---|-----|--|
| <pre> 5  scanf("%d",&amp;size); 6  int arr[size]; 7  printf("Enter %d numbers: ", size); 8  for(i=0; i&lt;size; i++){ 9      scanf("%d", &amp;arr[i]); 10 } 11 printf("The array element are: \n"); 12 for(i=0;i&lt;size;i++){ 13     printf("%d ", arr[i]); 14 } 15 printf("\nThe maximum number is: "); 16 int maximum = arr[0]; 17 for(i=1; i&lt;size; i++){ 18     if(arr[i] &gt; maximum) 19     { 20         maximum=arr[i]; 21     } 22 } </pre> |     | <pre> Enter any array size: 5 Enter 5 numbers: 10 20 50 30 40 The array element are: 10 20 50 30 40 The maximum number is: 50 === Code Execution Successful === </pre> |

Date: 14-09-2024

Problem No: 04

Problem Statement: Find the Even Numbers in the Array.

```

#include <stdio.h>

int main() {
    int i, size;
    printf("Enter any array size: ");
    scanf("%d",&size);
    int arr[size];
    printf("Enter %d numbers: ", size);

```

```

for(i=0; i<size; i++){
scanf("%d", &arr[i]);
}

printf("The array elements are: \n");

for(i=0; i<size; i++){
    printf("%d ", arr[i]);
}

printf("\nThe even numbers are: ");

for(i=0; i<size; i++){
if(arr[i]%2==0){
    printf("%d ", arr[i]);
}
}

return 0;
}

```

| main.c   | Output   |
|--|--|
| <pre> 2 ▸ int main(){ 3   int i, size; 4   printf("Enter any array size: "); 5   scanf("%d",&amp;size); 6   int arr[size]; 7   printf("Enter %d numbers: ", size); 8 ▸ for(i=0; i&lt;size; i++){ 9   scanf("%d", &amp;arr[i]); 10  } 11 12  printf("The array elements are: \n"); 13 ▸   for(i=0;i&lt;size;i++){ 14       printf("%d ", arr[i]); 15       } 16  printf("\nThe even numbers are: "); 17 ▸  for(i=0; i&lt;size; i++){ 18 ▸  if(arr[i]%2==0){ 19       printf("%d ", arr[i]); 20       } 21  } </pre> | <pre> Enter any array size: 5 Enter 5 numbers: 1 2 3 4 5 The array elements are: 1 2 3 4 5 The even numbers are: 2 4  === Code Execution Successful === </pre> |

Date: 14-09-2024

Problem No: 05

Problem Statement: Find the Odd Numbers in the Array.

```

#include <stdio.h>

int main(){
    int i, size;

    printf("Enter any array size: ");
    scanf("%d",&size);

    int arr[size];



    printf("Enter %d numbers: ", size);
    for(i=0; i<size; i++){
        scanf("%d", &arr[i]);
    }

    printf("The array elements are: \n");
        for(i=0; i<size; i++){
            printf("%d ", arr[i]);
        }

    printf("\nThe odd numbers are: ");
    for(i=0; i<size; i++){
        if(arr[i]%2!=0){
            printf("%d ", arr[i]);
        }
    }

    return 0;
}

```

|   |   |     |   |
|---|---|-----|---|
| main.c  |    Share | Run | Output  |
| <pre>2- int main(){ 3  int i, size; 4  printf("Enter any array size: "); 5  scanf("%d",&amp;size); 6  int arr[size]; 7 8  printf("Enter %d numbers: ", size); 9- for(i=0; i&lt;size; i++){ 10 scanf("%d", &amp;arr[i]); 11 } 12 printf("The array elements are: \n"); 13- for(i=0; i&lt;size; i++){ 14     printf("%d ", arr[i]); 15 } 16 printf("\nThe odd numbers are: "); 17- for(i=0; i&lt;size; i++){ 18- if(arr[i]%2!=0){ 19     printf("%d ", arr[i]);</pre> |   |     | <pre>Enter any array size: 5 Enter 5 numbers: 1 2 3 4 5 The array elements are: 1 2 3 4 5 The odd numbers are: 1 3 5  === Code Execution Successful ===</pre> |

Date: 14-09-2024

Problem No: 06

Problem Statement: Find the Positive Numbers in the Array.

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

```
int main(){
```

```
int i, size;
```

```
printf("Enter any array size: ");
```

```
scanf("%d",&size);
```

```
int arr[size];
```

```
printf("Enter %d numbers: ", size);
```

```
for(i=0; i<size; i++){
```

```
scanf("%d", &arr[i]);
```

```
}
```

```
printf("The array elements are: \n");
```

```
for(i=0; i<size; i++){
```

```
    printf("%d ", arr[i]);
```

```
}
```

```
printf("\nThe positive numbers are: ");
```



```

for(i=0; i<size; i++){
    if(arr[i] >= 0){
        printf("%d ", arr[i]);
    }
}

return 0;
}

```

| main.c   | Run | Output  |
|--|-----|---|
| <pre> 2 int main(){ 3 int i, size; 4 printf("Enter any array size: "); 5 scanf("%d",&amp;size); 6 int arr[size]; 7 8 printf("Enter %d numbers: ", size); 9 for(i=0; i&lt;size; i++){ 10 scanf("%d", &amp;arr[i]); 11 } 12 printf("The array elements are: \n"); 13 for(i=0; i&lt;size; i++){ 14     printf("%d ", arr[i]); 15 } 16 printf("\nThe positive numbers are: "); 17 for(i=0; i&lt;size; i++){ 18 if(arr[i]&gt;=0){ 19     printf("%d ", arr[i]); 20 } </pre> |     | <pre> Enter any array size: 5 Enter 5 numbers: -2 -1 6 0 -8 The array elements are: -2 -1 6 0 -8 The positive numbers are: 6 0 </pre> |
|  |     | <pre> === Code Execution Successful === </pre>  |

Date: 14-09-2024

Problem No: 07

Problem Statement: Find the Negative Numbers in the Array.

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

```
int main(){
```

```
int i, size;
```

```
printf("Enter any array size: ");
```

```
scanf("%d",&size);
```

```
int arr[size];
```

```
printf("Enter %d numbers: ", size);
```

```
for(i=0; i<size; i++){
```

```

scanf("%d", &arr[i]);
}
printf("The array elements are: \n");
for(i=0; i<size; i++){
    printf("%d ", arr[i]);
}
printf("\nThe negative numbers are: ");
for(i=0; i<size; i++){
    if(arr[i]<0){
        printf("%d ", arr[i]);
    }
}
return 0;
}

```

| main.c   | Run            | Output  |
|--|----------------|---|
| <pre> 3  int i, size; 4  printf("Enter any array size: "); 5  scanf("%d",&amp;size); 6  int arr[size]; 7 8  printf("Enter %d numbers: ", size); 9  for(i=0; i&lt;size; i++){ 10     scanf("%d", &amp;arr[i]); 11 } 12 printf("The array elements are: \n"); 13     for(i=0;i&lt;size;i++){ 14         printf("%d ", arr[i]); 15     } 16 printf("\nThe negative numbers are: "); 17     for(i=0; i&lt;size; i++){ 18     if(arr[i]&lt;0){ 19         printf("%d ", arr[i]); </pre> | <div>Run</div> | <pre> Enter any array size: 5 Enter 5 numbers: -2 -6 0 1 -8 The array elements are: -2 -6 0 1 -8 The negative numbers are: -2 -6 -8  === Code Execution Successful === </pre> |

Date: **21-09-2024**

Problem No: **08**

Problem Statement: **Insert a Number in the Array.**

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

```
int main()
```

```
{
```

```

int arr[100], i, size, position, value;
printf("Enter any array aize: ");
scanf("%d", &size);
printf("Enter %d Array Element: ", size);
for(i=0;i<size;i++){
    scanf("%d", &arr[i]);
}
for(i=0; i<size; i++)
{
    printf("%d ", arr[i]);
}
printf("\nEnter the position you want to insert: ");
scanf("%d", &position);
printf("Enter the value you want to insert: ");
scanf("%d", &value);
for(i=size; i>=position; i--){
    arr[i]=arr[i-1];
}
arr[position-1]=value;
printf("The Array Element After Insertion: \n");
for(i=0; i<size+1; i++){
    printf("%d ", arr[i]);
}
return 0;
}

```

```
main.c  [Icons] [Share] [Run] Output
7   printf("Enter %d Array Element: ", size);
8   for(i=0;i<size;i++){
9       scanf("%d", &arr[i]);
10  }
11  for(i=0;i<size;i++)
12  {
13      printf("%d ", arr[i]);
14  }
15  printf("\nEnter the position you want to insert: ");
16  scanf("%d", &position);
17  printf("Enter the value you want to insert: ");
18  scanf("%d", &value);
19  for(i=size;i>=position;i--){
20      arr[i]=arr[i-1];
21  }
22  arr[position-1]=value;
23  printf("The Array Element After Insertion: \n");
24  for(i=0;i<size+1;i++){
25      printf("%d ", arr[i]);
26  }
```

```
Enter any array aize: 5
Enter 5 Array Element: 1 2 3 4 5
1 2 3 4 5
Enter the position you want to insert: 3
Enter the value you want to insert: 88
The Array Element After Insertion:
1 2 88 3 4 5

=== Code Execution Successful ===
```

Date: 19-10-2024

Problem No: 09

Problem Statement: **LinkedList Creation Basic Concept and Manual Code.**

```
#include <stdio.h>

#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

void display(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;
```

```

struct Node* second = NULL;
struct Node* third = NULL;
head = (struct Node*)malloc(sizeof(struct Node));
second = (struct Node*)malloc(sizeof(struct Node));
third = (struct Node*)malloc(sizeof(struct Node));
head->data = 1;
head->next = second;
second->data = 2;
second->next = third;
third->data = 3;
third->next = NULL;
display(head);
return 0;
}

```

```

main.c
15 int main() {
16     struct Node* head = NULL;
17     struct Node* second = NULL;
18     struct Node* third = NULL;
19
20     head = (struct Node*)malloc(sizeof(struct Node));
21     second = (struct Node*)malloc(sizeof(struct Node));
22     third = (struct Node*)malloc(sizeof(struct Node));
23
24     head->data = 1;
25     head->next = second;
26
27     second->data = 2;
28     second->next = third;
29
30     third->data = 3;
31     third->next = NULL;
32
33     display(head);
34 }

```

Output

```

1 -> 2 -> 3 -> NULL
=== Code Execution Successful ===

```

Date: 19-10-2024

Problem No: 10

Problem Statement: Binary searching algorithm in an Array.

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

```
int main() {
```

```

int i, minimum, maximum, middle, size, search, arr[100];

printf("Enter array elements size: ");
scanf("%d", &size);

printf("Enter %d integer elements: ", size);

for (i=0; i<size; i++){
    scanf("%d",&arr[i]);
}

printf("The array elements are: \n");
for(i=0; i<size; i++){
    printf("%d ", arr[i]);
}

printf("\nEnter the value to search: ");
scanf("%d", &search);
minimum = 0;
maximum = size - 1;
middle = (minimum+maximum)/2;
while (minimum <= maximum) {
    if (arr[middle] < search)
        minimum = middle + 1;
    else if (arr[middle] == search) {
        printf("Element %d Found at index: %d\n", search, middle);
        printf("Element %d Found at position: %d\n", search, middle+1);
        break;
    }
    else
        maximum = middle - 1;
    middle = (minimum + maximum)/2;
}
if(minimum>maximum){

```

```

printf("Element not found.");
}
return 0;
}

```

The screenshot shows a C code editor with a file named 'main.c'. The code implements a binary search algorithm. A red box highlights the main search loop from line 24 to 37. The output window on the right shows the program's execution: it prompts for the array size (5), the elements (10 20 30 40 50), and the search value (50). It then reports the element was found at index 4 and position 5. The execution is successful.

```

main.c
24- while (minimum <= maximum) {
25-
26-     if (arr[middle] < search)
27-         minimum = middle + 1;
28-     else if (arr[middle] == search) {
29-         printf("Element %d Found at index: %d\n", search, middle);
30-         printf("Element %d Found at position: %d\n", search, middle+1);
31-         break;
32-     }
33-     else
34-         maximum = middle - 1;
35-
36-     middle = (minimum + maximum)/2;
37- }
38-
39- if(minimum>maximum){
40-     printf("Element not found.");
41- }

```

Output

```

Enter array elements size: 5
Enter 5 integer elements: 10 20 30 40 50
The array elements are:
10 20 30 40 50
Enter the value to search: 50
Element 50 Found at index: 4
Element 50 Found at position: 5

=== Code Execution Successful ===

```

Date: 19-10-2024

Problem No: 11

Problem Statement: **Linear Search in Linked List with Multiple Occurrences.**

```

#include <stdio.h>

#include <stdlib.h>

struct Node {
    int data;
    struct Node* next; };

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode; }

```

```

void appendNode(struct Node** head, int data) {

struct Node* newNode = createNode(data);

if (*head == NULL) {

    *head = newNode;

return; }

struct Node* temp = *head;
while (temp->next != NULL) {
    temp = temp->next;
}
temp->next = newNode;

}

void linearSearch(struct Node* head, int target) {

struct Node* temp = head;

int position = 0;

int found = 0;

while (temp != NULL) {
    if (temp->data == target) {
        printf("Found %d at node: %d\n", target, position+1);
        found = 1;
    }
    temp = temp->next;
    position++;
}

if (!found) {
    printf("%d not found in the list.\n", target);
}
}

void printList(struct Node* head) {

struct Node* temp = head; while (temp != NULL) {

printf("%d -> ", temp->data); temp = temp->next; }

printf("NULL\n"); }

int main() {

struct Node* head = NULL;

```



```

int size, value, target;

printf("Enter the linked list size: ");
scanf("%d", &size);

printf("Enter %d Linked list elements: ", size);
for (int i = 0; i < size; i++) {
    scanf("%d", &value);
    appendNode(&head, value);
}

printf("The linked list elements are: \n");
printList(head);

printf("Enter the value to search: ");
scanf("%d", &target);

printf("Searching for %d in the list...\n", target);
linearSearch(head, target);

return 0;
}

```

The screenshot shows a C code editor with a file named 'main.c'. The code defines a 'linearSearch' function that traverses a linked list to find a target value. The function takes a 'head' pointer and an 'int target' as arguments. It initializes a 'temp' pointer to 'head', a 'position' counter to 0, and a 'found' flag to 0. A 'while' loop continues until 'temp' is NULL. Inside the loop, it checks if 'temp->data' equals the 'target'. If true, it prints the target and its position (starting from 1), sets 'found' to 1, and moves 'temp' to 'temp->next' and 'position' to 'position++'. After the loop, if 'found' is still 0, it prints a message indicating the target was not found. The output window on the right shows the execution results: the user entered a size of 5 and elements 10, 20, 30, 40, 50. The list is printed as '10 -> 20 -> 30 -> 40 -> 50 -> NULL'. The user then entered 40 as the target, and the output shows 'Found 40 at node: 4'. A red box highlights the 'linearSearch' function in the code, and a red arrow points from this box to the corresponding output in the output window. The output window also displays '=== Code Execution Successful ==='.

```

main.c
29
30 void linearSearch(struct Node* head, int target) {
31     struct Node* temp = head;
32     int position = 0;
33     int found = 0;
34
35     while (temp != NULL) {
36         if (temp->data == target) {
37             printf("Found %d at node: %d\n", target, position+1);
38             found = 1;
39         }
40         temp = temp->next;
41         position++;
42     }
43
44     if (!found) {
45         printf("%d not found in the list.\n", target);
46     }
47 }

```

Output

```

Enter the linked list size: 5
Enter 5 Linked list elements: 10 20 30 40 50
The linked list elements are:
10 -> 20 -> 30 -> 40 -> 50 -> NULL
Enter the value to search: 40
Searching for 40 in the list...
Found 40 at node: 4

=== Code Execution Successful ===

```

Date: 26-10-2024

Problem No: 12

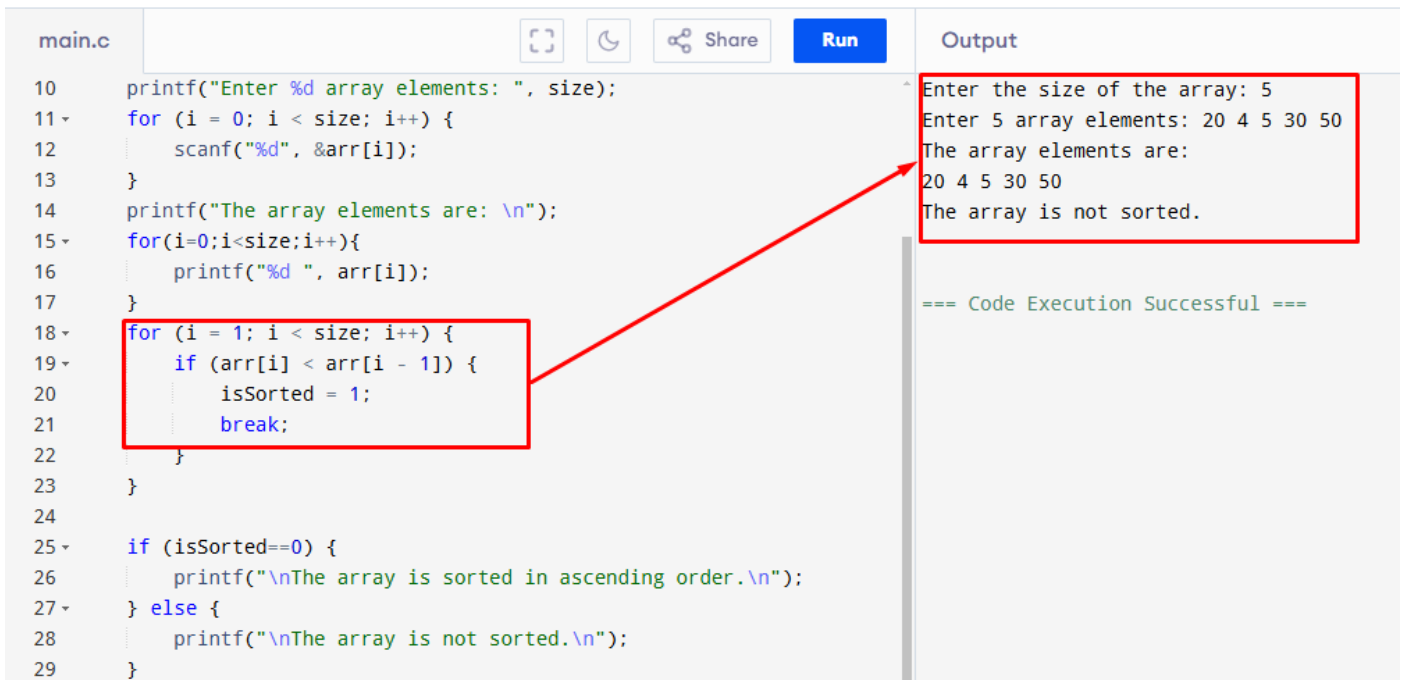
Problem Statement: Check an array is sorted or not.

```

#include <stdio.h>

int main() {
    int i, size, isSorted = 0;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter %d array elements: ", size);
    for (i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    printf("The array elements are: \n");
    for(i=0;i<size;i++){
        printf("%d ", arr[i]);
    }
    for (i = 1; i < size; i++) {
        if (arr[i] < arr[i - 1]) {
            isSorted = 1;
            break;
        }
    }
    if (isSorted==0) {
        printf("\nThe array is sorted in ascending order.\n");
    } else {
        printf("\nThe array is not sorted.\n");
    }
    return 0;
}

```



```
main.c
10 printf("Enter %d array elements: ", size);
11 for (i = 0; i < size; i++) {
12     scanf("%d", &arr[i]);
13 }
14 printf("The array elements are: \n");
15 for(i=0;i<size;i++){
16     printf("%d ", arr[i]);
17 }
18 for (i = 1; i < size; i++) {
19     if (arr[i] < arr[i - 1]) {
20         isSorted = 1;
21         break;
22     }
23 }
24
25 if (isSorted==0) {
26     printf("\nThe array is sorted in ascending order.\n");
27 } else {
28     printf("\nThe array is not sorted.\n");
29 }
```

Output

```
Enter the size of the array: 5
Enter 5 array elements: 20 4 5 30 50
The array elements are:
20 4 5 30 50
The array is not sorted.

=== Code Execution Successful ===
```

Date: 26-10-2024

Problem No: 13

Problem Statement: Split an Array into Two Parts Even First Part and Odd Elements Second Part.

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

```
int main() {
```

```
    int i, size;
```

```
    printf("Enter the size of the array: ");
```

```
    scanf("%d", &size);
```

```
    int arr[size], even[size], odd[size];
```

```
    int evenCount = 0, oddCount = 0;
```

```
    printf("Enter %d array elements: ", size);
```

```
    for (i = 0; i < size; i++) {
```

```
        scanf("%d", &arr[i]);
```

```
        if (arr[i] % 2 == 0) {
```

```

        even[evenCount++] = arr[i];
    } else {
        odd[oddCount++] = arr[i];
    }
}

printf("The array element are: \n");
for(i=0;i<size;i++){
    printf("%d ", arr[i]);
}

printf("\nEven part of the array are:\n");
for (i = 0; i < evenCount; i++) {
    printf("%d ", even[i]);
}

printf("\nOdd part of the array are:\n");
for (i = 0; i < oddCount; i++) {
    printf("%d ", odd[i]);
}

return 0;
}

```

The screenshot shows a C program in an IDE. The code in `main.c` is as follows:

```

main.c
~
7  int arr[size], even[size], odd[size];
8  int evenCount = 0, oddCount = 0;
9
10 printf("Enter %d array elements: ", size);
11 for (i = 0; i < size; i++) {
12     scanf("%d", &arr[i]);
13
14     if (arr[i] % 2 == 0) {
15         even[evenCount++] = arr[i];
16     } else {
17         odd[oddCount++] = arr[i];
18     }
19 }
20
21 printf("The array element are: \n");
22 for(i=0;i<size;i++){
23     printf("%d ", arr[i]);
24 }
25

```

The output of the program is shown on the right:

```

Output
Enter the size of the array: 10
Enter 10 array elements: 1 2 3 4 5 6 7 8 9 10
The array element are:
1 2 3 4 5 6 7 8 9 10
Even part of the array are:
2 4 6 8 10
Odd part of the array are:
1 3 5 7 9

=== Code Execution Successful ===

```

A red box highlights the input section of the code (lines 10-19) and the corresponding output (lines 1-9). A red arrow points from the input section of the code to the output section.

Date: **01-11-2024**

Problem No: **14**

Problem Statement: **Implement a Stack to Support Efficient Creation, Data insertion, and Deletion Operations.**

```
#include<stdio.h>

int top = -1;
int stack[5];

int isFull(){
    if (top==4){
        return 1;
    }else{
        return 0;
    } }

int isEmpty(){
    if(top == -1){
        return 1;
    }else{
        return 0;
    }
}

void push(int data){
    if (isFull()){
        printf("Stack Overflow!\n");
        return;
    }
    top = top+1;
    stack[top] = data;
}

void pop(){
    if(isEmpty()) {
        printf("Stack is empty!\n");
```

```

        return;
    }
    printf("Popped: %d\n",stack[top]);
    top=top-1;
}

void print(){
    if(isEmpty()){
        printf("Empty Stack");
        return;
    }
    printf("Your Current Stack:\n");
    for(int i=top;i>=0;i--){
        printf("%d\n", stack[i]);
    }
}

int main(){
    push(1);
    push(2);
    push(3);
    push(4);
    push(5);
    print();
    pop();
    print();
    pop();
    print();
    push(6);
    print();
    pop();
    pop();
    pop();
    pop();

```

```

    print();
return 0;
}

```

The screenshot shows a C program in a code editor with the following code:

```

54 int main()
55 {
56     push(1);
57     push(2);
58     push(3);
59     push(4);
60     push(5);
61     print();
62     pop();
63     print();
64     pop();
65     print();
66     push(6);
67     print();
68     pop();
69     pop();
70     pop();
71     pop();
72     print();
73
74 return 0;

```

The output window shows the following sequence of events:

```

Your Current Stack:
5
4
3
2
1
Popped: 5
Your Current Stack:
4
3
2
1
Popped: 4
Your Current Stack:
3
2
1
Your Current Stack:
6
3
2

```

A red box highlights the code from line 54 to 74, and another red box highlights the output from "Your Current Stack: 5" to "6 3 2". A red arrow points from the first `print()` call (line 61) to the first "Your Current Stack:" output.

Date: 02-11-2024

Problem No: 15

Problem Statement: Reverse an Array Using a Stack.

```

#include<stdio.h>

#define stack_size 100

int top = -1;

int stack[stack_size];

int isFull(){
    if (top==stack_size-1){
        return 1;
    }else{
        return 0;
    }
}

```

```

    } }
int isEmpty(){
    if(top == -1){
        return 1;
    }
    else{
        return 0;
    }
}
void push(int data){
    if (isFull()){
        printf("Stack Overflow!\n");
        return;
    }
    top++;
    stack[top] = data;
}
int pop(){
    if(isEmpty()){
        printf("Stack is empty!\n");
        return 1;
    }
    printf("Popped element: %d\n",stack[top]);
    return stack[top--];
}
int main(){
    int i,size;
    printf("Enter array element size: ");
    scanf("%d",&size);
    int arr[size];
    printf("Enter %d array element: ", size);
    for(i=0;i<size;i++){

```



```

scanf("%d",&arr[i]);
}
printf("\nArray element are: \n");
for(i=0;i<size;i++){
    printf("%d ",arr[i]);
}
printf("\n\n");
for(i=0;i<size;i++){
    push(arr[i]);
}
for (int i=0; i<size; i++) {
    arr[i] = pop();
}
printf("\nReverse Array element are: \n");
for(i=0;i<size;i++){
    printf("%d ",arr[i]);
}
return 0;
}

```

The screenshot shows a C program in a code editor with the following code:

```

main.c
55 printf("\nArray element are: \n");
56 for(i=0;i<size;i++){
57     printf("%d ",arr[i]);
58 }
59
60 printf("\n\n");
61 for(i=0;i<size;i++){
62     push(arr[i]);
63 }
64
65 for (int i=0; i<size; i++) {
66     arr[i] = pop();
67 }
68
69
70 printf("\nReverse Array element are: \n");
71 for(i=0;i<size;i++){
72     printf("%d ",arr[i]);
73 }
74

```

The output of the program is shown in the terminal:

```

Enter array element size: 5
Enter 5 array element: 1 2 3 4 5

Array element are:
1 2 3 4 5

Popped element: 5
Popped element: 4
Popped element: 3
Popped element: 2
Popped element: 1

Reverse Array element are:
5 4 3 2 1

=== Code Execution Successful ===

```

A red box highlights the push and pop logic in the code (lines 60-67), and an arrow points from this box to the corresponding output in the terminal.

Date: **09-11-2024**

Problem No: **16**

Problem Statement: **Sum of Two Numbers Using Pointers.**

```
#include <stdio.h>

int main(){

int num1, num2, sum; int *ptr1, *ptr2;

printf("Enter the first number: ");
scanf("%d", &num1);

printf("Enter the second number: ");
scanf("%d", &num2);

ptr1 = &num1;
ptr2 = &num2;

sum = *ptr1 + *ptr2;

printf("The sum of %d and %d is: %d\n", *ptr1, *ptr2, sum);

return 0;

}
```

```
main.c
1 #include <stdio.h>
2 int main(){
3     int num1, num2, sum;
4     int *ptr1, *ptr2;
5
6     printf("Enter the first number: ");
7     scanf("%d", &num1);
8
9     printf("Enter the second number: ");
10    scanf("%d", &num2);
11
12    ptr1 = &num1;
13    ptr2 = &num2;
14
15    sum = *ptr1 + *ptr2;
16
17    printf("The sum of %d and %d is: %d\n", *ptr1, *ptr2, sum);
18
19    return 0;
20 }
```

Output

```
Enter the first number: 10
Enter the second number: 5
The sum of 10 and 5 is: 15

=== Code Execution Successful ===
```

Date: 09-11-2024

Problem No: 17

Problem Statement: Swap Two Numbers Using Pointers With Temp Variable.

```
#include <stdio.h>

int main(){

int A=20; int B=10;

printf("\nBefore Swaping A = %d, B = %d.\n",A,B);

int *x = &A;

int *y = &B;

int temp = *x;

*x = *y;

*y = temp;

printf("\nAfter Swaping A = %d, B = %d.\n",A,B);

return 0;

}
```

```
main.c
1 #include <stdio.h>
2 int main(){
3     int A=20;
4     int B=10;
5     printf("\nBefore Swaping A = %d, B = %d.\n",A,B);
6
7     int *x = &A;
8     int *y = &B;
9     int temp = *x;
10    *x = *y;
11    *y = temp;
12
13    printf("\nAfter Swaping A = %d, B = %d.\n",A,B);
14
15    return 0;
16 }
```

Output

Before Swaping A = 20, B = 10.

After Swaping A = 10, B = 20.

=== Code Execution Successful ===

Date: 09-11-2024

Problem No: 18

Problem Statement: Swap Two Numbers Using Pointers Without Temp Variable.

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

```
int main(){
```

```
int A=20;
```

```
int B=10;
```

```
printf("\nBefore Swaping A = %d, B = %d.\n",A,B);
```

```
int *x = &A;
```

```
int *y = &B;
```

```
*x = *x+*y;
```

```
*y = *x-*y;
```

```
*x = *x-*y;
```

```
printf("\nAfter Swaping A = %d, B = %d.\n",A,B);
```

```
return 0;
```

```
}
```

The screenshot shows a C code editor with a file named `main.c`. The code implements a swap function using pointer arithmetic. A red box highlights the pointer declarations and the swap logic, with a red arrow pointing from it to the output. The output shows the state of variables A and B before and after the swap, followed by a success message.

```
main.c
1 #include <stdio.h>
2 int main(){
3     int A=20;
4     int B=10;
5     printf("\nBefore Swaping A = %d, B = %d.\n",A,B);
6
7     int *x = &A;
8     int *y = &B;
9
10    *x = *x+*y;
11    *y = *x-*y;
12    *x = *x-*y;
13
14    printf("\nAfter Swaping A = %d, B = %d.\n",A,B);
15
16    return 0;
17 }
18
```

Output

Before Swaping A = 20, B = 10.  
After Swaping A = 10, B = 20.

=== Code Execution Successful ===

Date: 26-11-2024

Problem No: 19

Problem Statement: Implement a Queue to Support Efficient Creation, Data Insertion, and Deletion Operations.

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

```
int front = -1;
```

```
int rear = -1;
```

```
int queue[5];
```

```
int isFull(){
```

```
    if (rear == 4){
```

```
        return 1;
```

```
    }else{
```

```
        return 0;
```

```
    } }
```

```
int isEmpty(){
```

```
    if (front == -1 || front > rear){
```

```
        return 1;
```

```
    }
```

```

else{
    return 0;
} }

void enqueue(int data){
    if (isFull()){
        printf("Queue Overflow!\n");
        return;
    }
    if (front == -1){
        front = 0;
    }
    rear = rear + 1;
    queue[rear] = data;
}

void dequeue(){
    if (isEmpty()) {
        printf("Queue is empty!\n");
        return;
    }
    printf("Dequeued: %d\n", queue[front]);
    front = front + 1;
}

void print(){
    if (isEmpty()){
        printf("Empty Queue\n");
        return;
    }
    printf("Your Current Queue:\n");
    for (int i = front; i <= rear; i++){
        printf("%d\n", queue[i]);
    }
}

```

```
int main(){
    enqueue(1);
    enqueue(2);
    enqueue(3);
    enqueue(4);
    enqueue(5);
    print();
    dequeue();
    print();
    dequeue();
    print();
    enqueue(6);
    print();
    dequeue();
    dequeue();
    dequeue();
    dequeue();
    print();
    enqueue(6);
    enqueue(7);
    enqueue(8);
    enqueue(9);
    enqueue(10);

    return 0;
}
```

The screenshot shows a C program in a code editor with the following code:

```
main.c
58 int main(){
59     enqueue(1);
60     enqueue(2);
61     enqueue(3);
62     enqueue(4);
63     enqueue(5);
64     print();
65     dequeue();
66     print();
67     dequeue();
68     print();
69     enqueue(6);
70     print();
71     dequeue();
72     dequeue();
73     dequeue();
74     dequeue();
75     print();
76     enqueue(6);
77     enqueue(7);
78 }
```

The output of the program is as follows:

```
Output
Your Current Queue:
1
2
3
4
5
Dequeued: 1
Your Current Queue:
2
3
4
5
Dequeued: 2
Your Current Queue:
3
4
5
Queue Overflow!
Your Current Queue:
3
4
```

A red box highlights the code and the output. An arrow points from the code to the output, indicating the flow of execution.

**The End**