HI-TECH INSTITUTE OF ENGINEERING AND TECHNOLOGY, GHAZIABAD

PRACTICAL FILE
DATA STRUCTURES USING C LANGUAGE
KCS-351
B.Tech (CS-AI) 2nd Year

SUBMITTED BY:

NAME: AMAN RAJPUT

ROLL NO.: 2102201530006

SUBMITTED TO:

Mr. DILIP KR. BHARTI

Dr.A.P.J. Abdul Kalam Technical University (U.P.)

INDEX

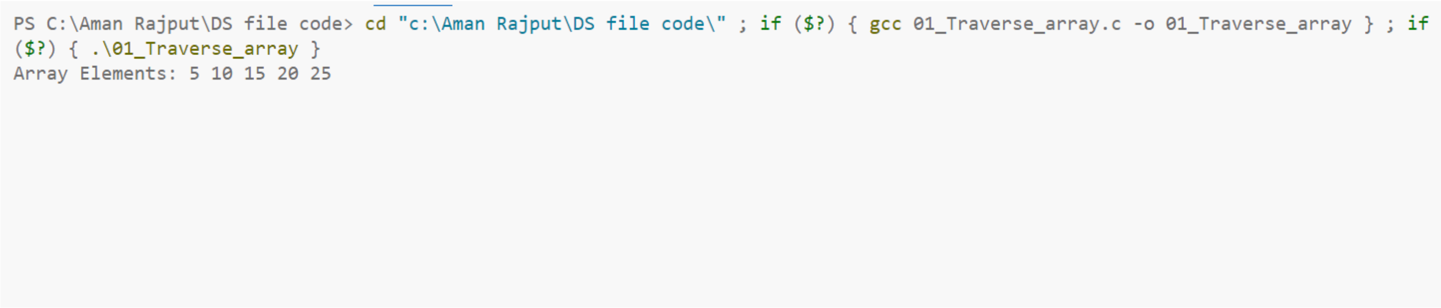
14	Write a C program to perform Merge Sort.	02-01-23		
15	Write a C program to implement Tower of Hanoi using recursion.	05-01-23		
16	Write a C program using RECURSION.	12-01-23		
17	Write a C program to perform FACTORIAL using ITERATIVE & RECURSION METHOD.	18-01-23		
18	Write a C program to perform FIBONACCI SERIES using the ITERATIVE method.	25-01-23		
19	Write a C program to perform TAIL RECURSION.	31-01-23		
20	Write a C program to perform FIBONACCI SERIES using the RECURSION method.	09-02-23		

EXPERIMENT NO-:01

Write a C program to Traverse Array Element

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

int main() {
    int arr[] = {5, 10, 15, 20, 25};
    int i, n = sizeof(arr) / sizeof(arr[0]);

    printf("Array Elements: ");

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

    printf("\n");

    getch();
    return 0;
}
```

OUTPUT -:



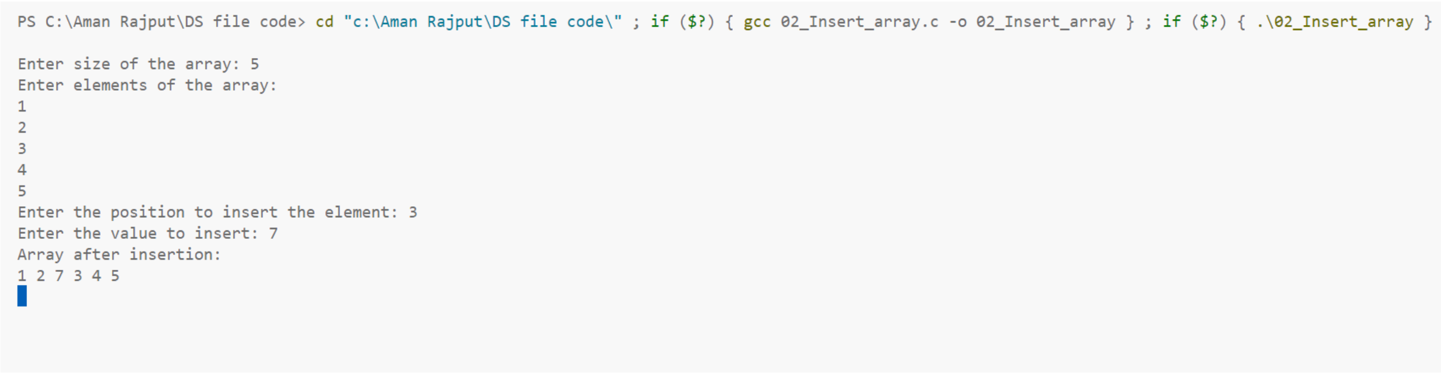
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 01_Traverse_array.c -o 01_Traverse_array } ; if
($?) { .\01_Traverse_array }
Array Elements: 5 10 15 20 25
```

EXPERIMENT NO-:02

Write a C program to Insert elements in an Array

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

int main() {
   int arr[100], i, size, pos, value;
   printf("Enter size of the array: ");
   scanf("%d", &size);
  printf("Enter elements of the array:\n");
   for(i = 0; i < size; i++) {
      scanf("%d", &arr[i]);
   }
   printf("Enter the position to insert the element: ");
   scanf("%d", &pos);
   printf("Enter the value to insert: ");
   scanf("%d", &value);
   if(pos <= 0 || pos > size + 1) {
      printf("Invalid position.\n");
   }
   else {
      for(i = size - 1; i >= pos - 1; i--) {
         arr[i+1] = arr[i];
      }
      arr[pos-1] = value;
      size++;
      printf("Array after insertion:\n");
      for(i = 0; i < size; i++) {
         printf("%d ", arr[i]);
      }
      printf("\n");
   }
   getch();
   return 0;
}
```

OUTPUT -:



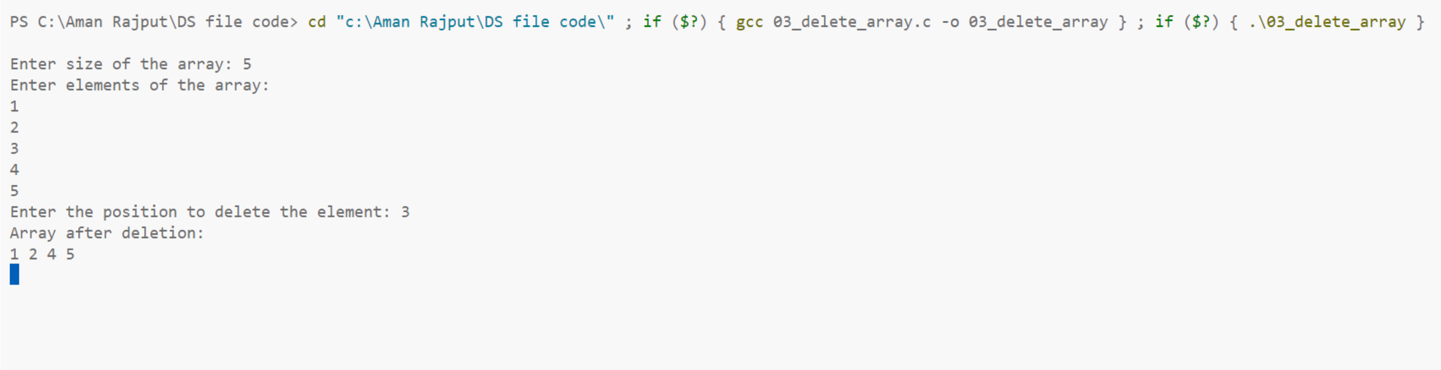
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 02_Insert_array.c -o 02_Insert_array } ; if ($?) { .\02_Insert_array }

Enter size of the array: 5
Enter elements of the array:
1
2
3
4
5
Enter the position to insert the element: 3
Enter the value to insert: 7
Array after insertion:
1 2 7 3 4 5
```

EXPERIMENT NO-:03

Write a C program to Delete elements of an Array

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

int main() {
    int arr[100], size, pos, i;

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

    printf("Enter elements of the array:\n");
    for(i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter the position to delete the element: ");
    scanf("%d", &pos);
    if(pos <= 0 || pos > size) {
        printf("Invalid position.\n");
    }
    else {
        for(i = pos - 1; i < size - 1; i++) {
            arr[i] = arr[i+1];
        }
        size--;
        printf("Array after deletion:\n");
        for(i = 0; i < size; i++) {
            printf("%d ", arr[i]);
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

OUTPUT -:



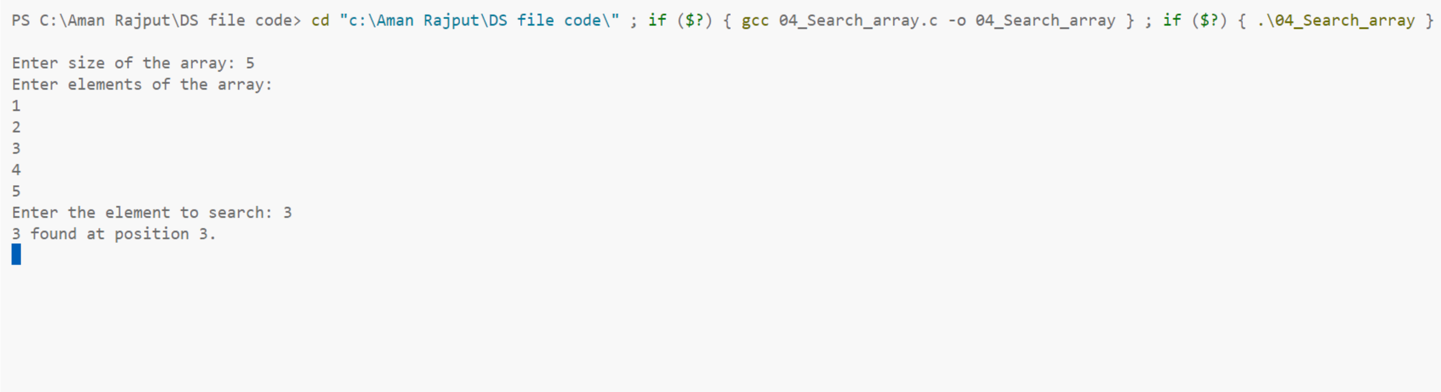
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 03_delete_array.c -o 03_delete_array } ; if ($?) { .\03_delete_array }

Enter size of the array: 5
Enter elements of the array:
1
2
3
4
5
Enter the position to delete the element: 3
Array after deletion:
1 2 4 5
```

EXPERIMENT NO-:04

Write a C program to Search elements in Array

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

int main() {
    int arr[100], i, size, search, found = 0;

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

    printf("Enter elements of the array:\n");
    for(i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the element to search: ");
    scanf("%d", &search);
    for(i = 0; i < size; i++) {
        if(arr[i] == search) {
            found = 1;
            printf("%d found at position %d.\n", search, i+1);
        }
    }
    if(!found) {
        printf("%d not found in the array.\n", search);
    }
    getch();
    return 0;
}
```

OUTPUT -:



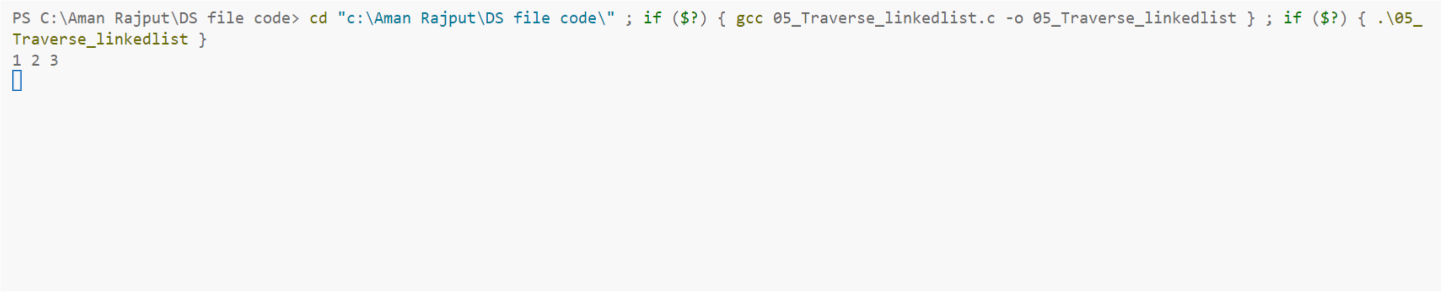
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 04_Search_array.c -o 04_Search_array } ; if ($?) { .\04_Search_array }

Enter size of the array: 5
Enter elements of the array:
1
2
3
4
5
Enter the element to search: 3
3 found at position 3.
```

EXPERIMENT NO-:05

Write a C program to Traverse Linked List

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct node {
   int data;
   struct node *next;
};
void traverseList(struct node *head) {
   struct node *current = head;
   while(current != NULL) {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\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;
   traverseList(head);
   getch();
   return 0;
}
```

OUTPUT -:



```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 05_Traverse_linkedlist.c -o 05_Traverse_linkedlist } ; if ($?) { .\05_
Traverse_linkedlist }
1 2 3
```

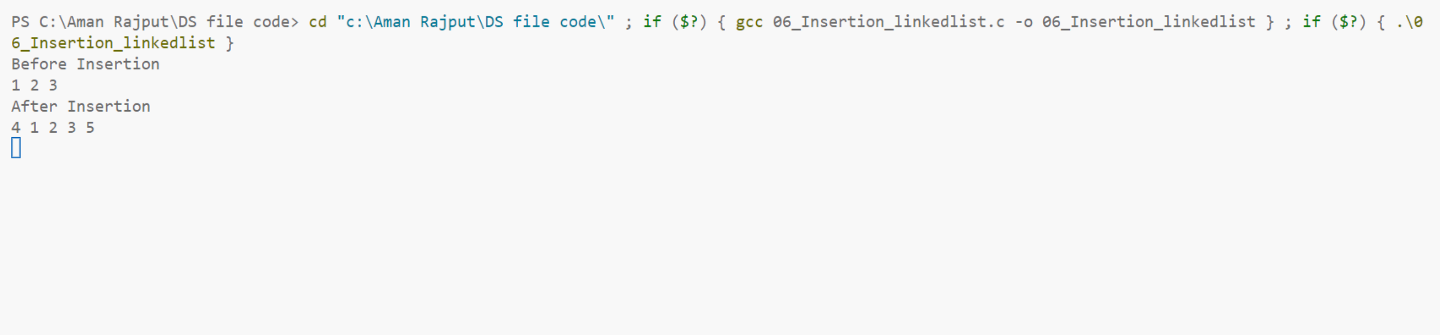
EXPERIMENT NO-:06

Write a C program to Insertion of Linked List

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

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

void traverseList(struct node *head) {
   struct node *current = head;

   while(current != NULL) {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\n");
}

struct node* insertatstart(struct node *head,int data){
   struct node * ptr =(struct node *)malloc(sizeof(struct node *));
   ptr->data=data;

   ptr->next=head;
   return ptr;
}

struct node* insertatend(struct node * head,int data){
struct node * ptr =(struct node *)malloc(sizeof(struct node *));
   ptr->data=data;
   struct node * p = head;

   while(p->next!=NULL){
      p=p->next;
   }

   p->next=ptr;
   ptr->next=NULL;
   return head;
}

int main() {
   struct node *head = NULL;

   head = (struct node *) malloc(sizeof(struct node));
   head->data = 1;
```

```
    struct node *second = (struct node *) malloc(sizeof(struct node));
    second->data = 2;

    struct node *third = (struct node *) malloc(sizeof(struct node));
    third->data = 3;

    head->next = second;
    second->next = third;
    third->next = NULL;

    printf("Before Insertion\n");
    traverseList(head);

    head=insertatstart(head, 4);
    head=insertatend(head, 5);

    printf("After Insertion\n");
    traverseList(head);

    return 0;
}
```

OUTPUT -:



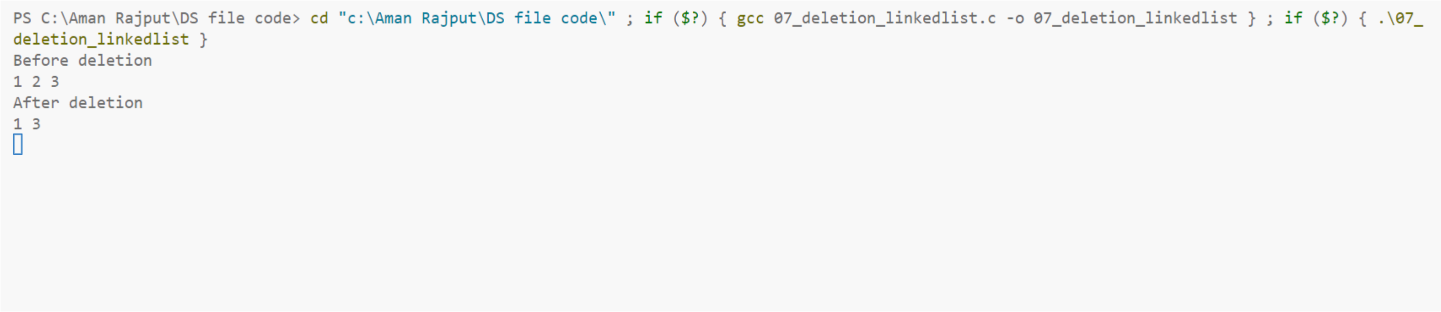
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 06_Insertion_linkedlist.c -o 06_Insertion_linkedlist } ; if ($?) { .\0
6_Insertion_linkedlist }
Before Insertion
1 2 3
After Insertion
4 1 2 3 5
```

EXPERIMENT NO-:07

Write a C program to Deletion of Linked List

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

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

void traverseList(struct node *head) {
   struct node *current = head;

   while(current != NULL) {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\n");
}

struct node * deletionatvalue(struct node *head,int value){
  struct node * p=head;
  struct node * q=head->next;
  if(head->data == value) {        //if value at first node
      struct node *temp = head;
      head = head->next;
      free(temp);

      return head;
   }
   else{
   while(q->data!=value && q->next!=NULL){
   p=p->next;
   q=q->next;
   }
   if(q->data==value){
   p->next=q->next;
   free(q);
   }
   return head;
   }
}

int main() {
   struct node *head = NULL;
   head = (struct node *) malloc(sizeof(struct node));
   head->data = 1;

   struct node *second = (struct node *) malloc(sizeof(struct node));
   second->data = 2;

   struct node *third = (struct node *) malloc(sizeof(struct node));
   third->data = 3;

   head->next = second;
   second->next = third;
   third->next = NULL;

   printf("Before deletion\n");
   traverseList(head);
   head=deletionatvalue(head, 2);
   printf("After deletion\n");
   traverseList(head);
   getch();
   return 0;
}
```

OUTPUT -:



```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 07_deletion_linkedlist.c -o 07_deletion_linkedlist } ; if ($?) { .\07_
deletion_linkedlist }
Before deletion
1 2 3
After deletion
1 3
```

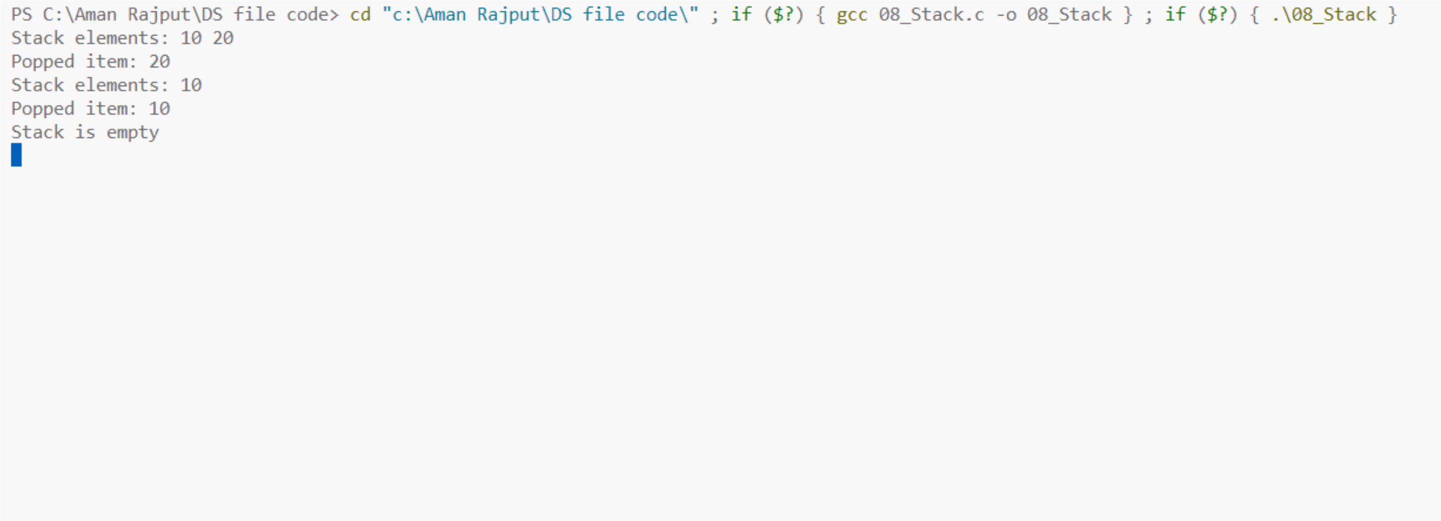
EXPERIMENT NO-:08

Write a C program to Perform PUSH and POP operation in Stack

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

#define SIZE 100

int stack[SIZE], top = -1;

void push(int item) {
   if(top >= SIZE - 1) {
      printf("Stack Overflow\n");
      return;
   }
   top++;
   stack[top] = item;
}

int pop() {
   if(top < 0) {
      printf("Stack Underflow\n");
      return -1;
   }
   int item = stack[top];
   top--;
   return item;
}

void display() {
   if(top < 0) {
      printf("Stack is empty\n");
      return;
   }
   printf("Stack elements: ");
   for(int i = 0; i <= top; i++) {
      printf("%d ", stack[i]);
   }
   printf("\n");
}

int main() {
   push(10);
   push(20);
   display();

   int item = pop();
```

```
    printf("Popped item: %d\n", item);
    display();
    item = pop();
    printf("Popped item: %d\n", item);

    display();
    getch();
    return 0;

}
```

OUTPUT -:



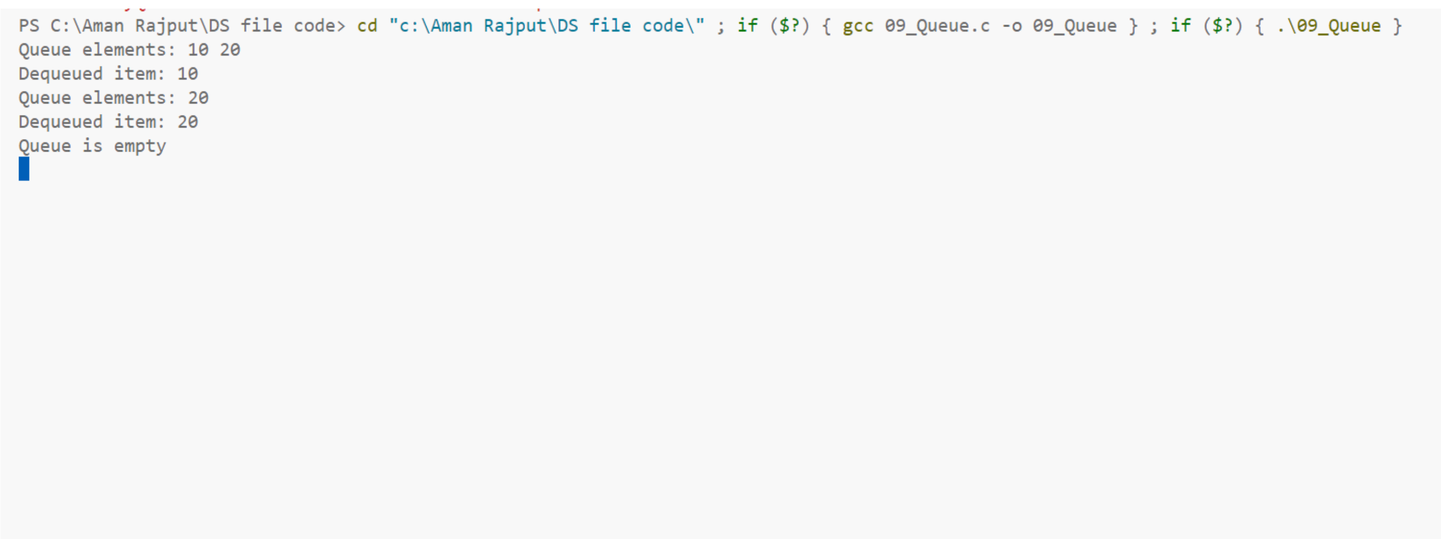
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 08_Stack.c -o 08_Stack } ; if ($?) { .\08_Stack }
Stack elements: 10 20
Popped item: 20
Stack elements: 10
Popped item: 10
Stack is empty
```

EXPERIMENT NO-:09

Write a C program to perform Queue Operation

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

#define SIZE 100

int queue[SIZE], front = -1, rear = -1;

void enqueue(int item) {
    if(rear >= SIZE - 1) {
        printf("Queue Overflow\n");
        return;
    }
    if(front == -1 && rear == -1) {
        front = rear = 0;
        queue[rear] = item;
    }
    else {
        rear++;
        queue[rear] = item;
    }
}

int dequeue() {
    if(front == -1 || front > rear) {
        printf("Queue Underflow\n");
        return -1;
    }
    int item = queue[front];
    front++;
    if(front > rear) {
        front = rear = -1;
    }
    return item;
}

void display() {
    if(front == -1 || front > rear) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements: ");
    for(int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    enqueue(10);
    enqueue(20);
    display();

    int item = dequeue();
    printf("Dequeued item: %d\n", item);
    display();

    item = dequeue();
    printf("Dequeued item: %d\n", item);
    display();
    getch();
    return 0;
}
```

OUTPUT -:



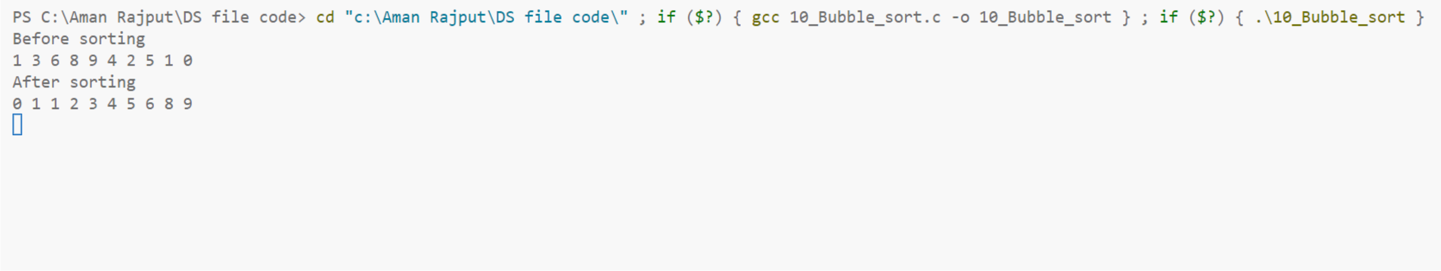
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 09_Queue.c -o 09_Queue } ; if ($?) { .\09_Queue }
Queue elements: 10 20
Dequeued item: 10
Queue elements: 20
Dequeued item: 20
Queue is empty
```

EXPERIMENT NO-:10

Write a C program to perform Bubble Sort

```
#include<stdio.h>
#include <conio.h>
void bubbleSort(int arr[], int size) {
    int i, j, temp;
    for(i = 0; i < size-1; i++) {
        for(j = 0; j < size-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
void display(int *arr,int size){
    int i;
    for(i=0;i<size;i++){
    printf("%d ",arr[i]);
    }
    printf("\n");
}
int main(){
    int arr[]={1,3,6,8,9,4,2,5,1,0};
    int size=10;
    printf("Before sorting\n");
    display(arr,size);
    bubbleSort(arr,size);
    printf("After sorting\n");
    display(arr,size);
    getch();
    return 0;
}
```

OUTPUT -:



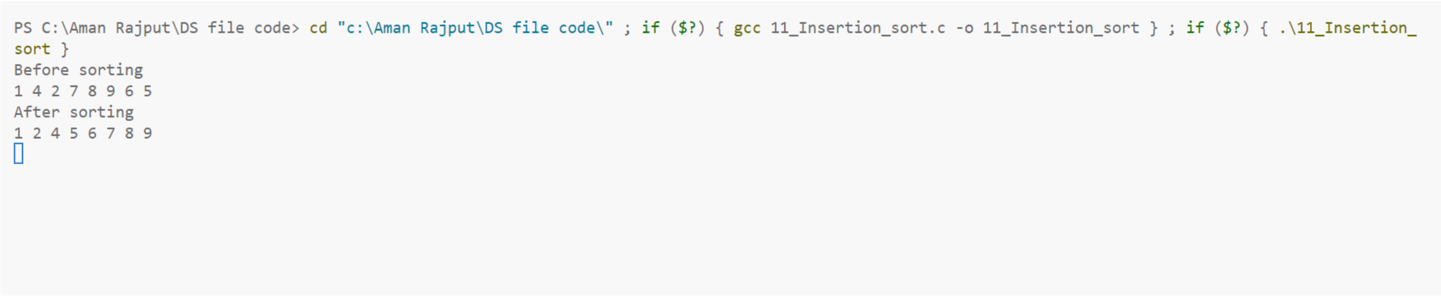
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 10_Bubble_sort.c -o 10_Bubble_sort } ; if ($?) { .\10_Bubble_sort }
Before sorting
1 3 6 8 9 4 2 5 1 0
After sorting
0 1 1 2 3 4 5 6 8 9
```

EXPERIMENT NO-:11

Write a C program to perform Insertion Sort

```
#include<stdio.h>
#include <conio.h>
void insertionSort(int *arr,int size){
    int key,i,j;
    for(i=1;i<size;i++){
        key=arr[i];
        j=i-1;
        while(j>=0 && arr[j]>key){
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=key;
    }
}
void display(int *arr, int size){
    int i;
    for(i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
}
int main(){
    int arr[]={1,4,2,7,8,9,6,5};
    int size=8;
    printf("Before sorting\n");
    display(arr,size);
    insertionSort(arr,size);
    printf("After sorting\n");
    display(arr,size);
    getch();
    return 0;
}
```

OUTPUT -:



```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 11_Insertion_sort.c -o 11_Insertion_sort } ; if ($?) { .\11_Insertion_
sort }
Before sorting
1 4 2 7 8 9 6 5
After sorting
1 2 4 5 6 7 8 9
```

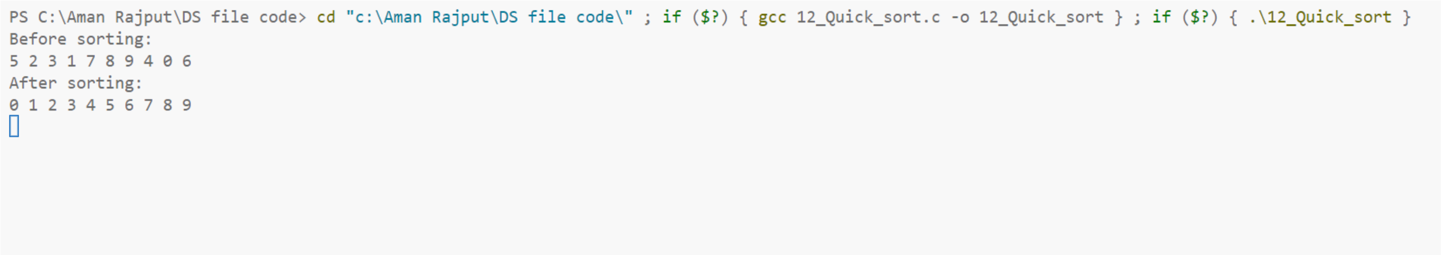
EXPERIMENT NO-:12

Write a C program to perform Quick Sort

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

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

int partition(int arr[], int low, int high){
  int i = low + 1;
  int j = high;
  int temp;

  do{
    while(arr[i]<=arr[low]){      //arr[low] is our pivot
      i++;
    }
    while(arr[j]>arr[low]){
      j--;
    }
    if(i<j){
      temp=arr[i];
      arr[i]=arr[j];
      arr[j]=temp;
    }
  }while(i<j);

    temp=arr[low];          //swap
    arr[low]=arr[j];
    arr[j]=temp;

  return j;

}

void quickSort(int arr[], int low, int high){
  int partitionIndex;

  if(low < high){
    partitionIndex = partition(arr,low,high);
    quickSort(arr,low,partitionIndex-1);
```

```
        quickSort(arr,partitionIndex+1,high);
    }
}

int main(){

    int arr[]={5, 2, 3, 1, 7, 8, 9, 4, 0, 6};
    int low=0;
    int size=10;

    display(arr,size);
    quickSort(arr,low,size-1);
    display(arr,size);
    getch();
    return 0;
}
```

OUTPUT -:



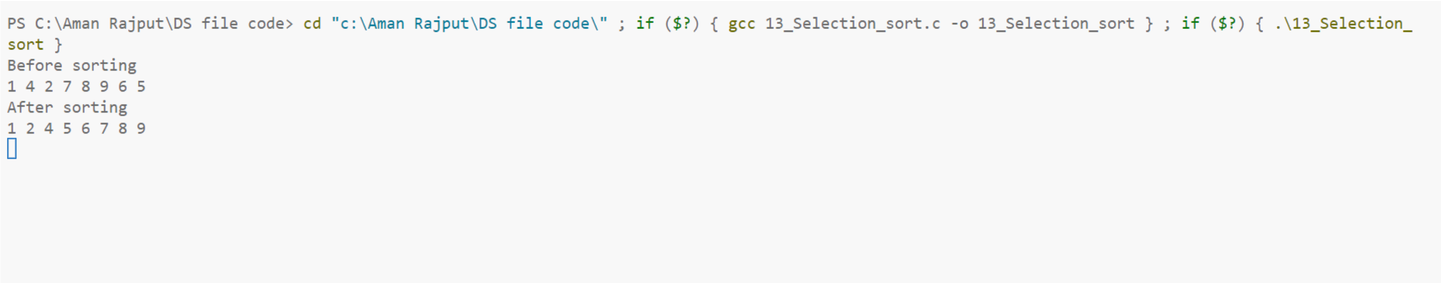
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 12_Quick_sort.c -o 12_Quick_sort } ; if ($?) { .\12_Quick_sort }
Before sorting:
5 2 3 1 7 8 9 4 0 6
After sorting:
0 1 2 3 4 5 6 7 8 9
```

EXPERIMENT NO-:13

Write a C program to perform Selection Sort

```
#include <stdio.h>
#include <conio.h>
void display(int *arr, int size){
    int i;
    for(i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
 }
void selectionSort(int *arr, int size){
    int i,j,indexSmall, temp;
    for(i=0 ; i<size-1 ; i++){        //loop 1 for passes(steps)
        indexSmall = i;
        for(j=i+1 ; j<size ; j++){                //loop 2 for comparison
            if(arr[j] < arr[indexSmall]){
                indexSmall = j;
            }
        }
        temp=arr[i];        //swap
        arr[i]=arr[indexSmall];
        arr[indexSmall]=temp;
    }
}
int main(){
    int arr[]={1,4,2,7,8,9,6,5};
    int size=8;
    printf("Before sorting\n");
    display(arr , size);
    selectionSort(arr , size);
    printf("After sorting\n");
    display(arr , size);
    getch();
    return 0;
}
```

OUTPUT -:



```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 13_Selection_sort.c -o 13_Selection_sort } ; if ($?) { .\13_Selection_
sort }
Before sorting
1 4 2 7 8 9 6 5
After sorting
1 2 4 5 6 7 8 9
```

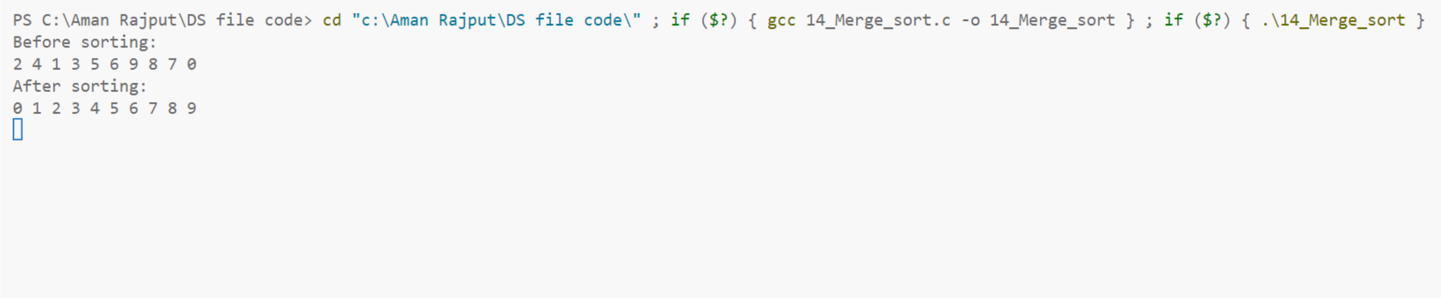
EXPERIMENT NO-:14

Write a C program to perform Merge Sort

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

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

void merge(int arr_1[], int low, int mid, int high){
    int i=low;
    int j=mid+1;
    int k=low;
    int arr_2[100];

    while(i<=mid && j<=high){
        if(arr_1[i]<arr_1[j]){
            arr_2[k]=arr_1[i];
            i++;
            k++;
        }
        else{
            arr_2[k]=arr_1[j];
            j++;
            k++;
        }
    }
    while(i<=mid){
        arr_2[k]=arr_1[i];
        i++;
        k++;
    }
    while(j<=high){
        arr_2[k]=arr_1[j];
        j++;
        k++;
    }

    for(i=low;i<=high;i++){
        arr_1[i]=arr_2[i];
    }
}
```

```
void mergeSort(int arr[], int low, int high){
    int mid;
    if(low < high){
        mid = (low + high) / 2;
        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);
        merge(arr, low, mid, high);
    }
}

int main()
{
    int arr[] = {2, 4, 1, 3, 5, 6, 9, 8, 7, 0};
    int size = 10;

    int low = 0;
    int high = 9;

    display(arr, size);
    mergeSort(arr, low, high);
    display(arr, size);
    getch();
    return 0;
}
```

OUTPUT -:



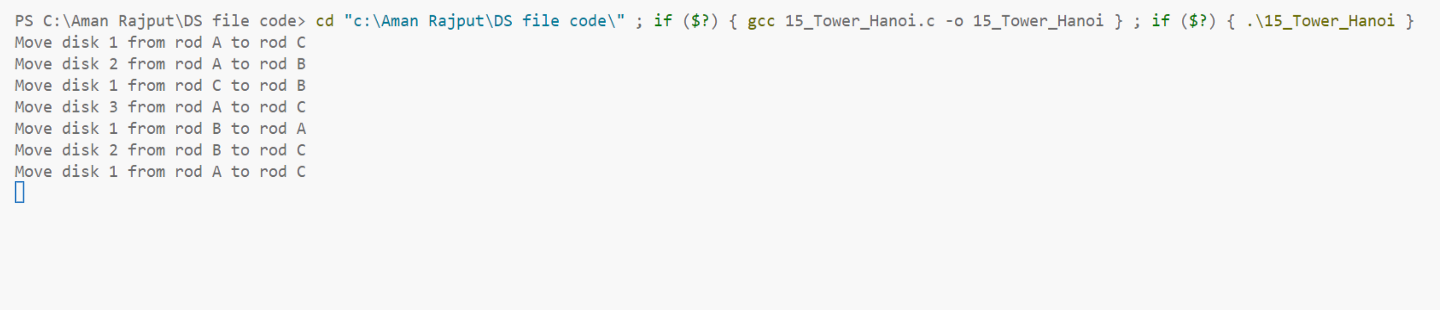
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 14_Merge_sort.c -o 14_Merge_sort } ; if ($?) { .\14_Merge_sort }
Before sorting:
2 4 1 3 5 6 9 8 7 0
After sorting:
0 1 2 3 4 5 6 7 8 9
```

EXPERIMENT NO-:15

Write a C program to implement Tower of Hanoi using recursion

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

void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod) {
   if(n == 1) {
      printf("Move disk 1 from rod %c to rod %c\n", from_rod, to_rod);
      return;
   }
   towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
   printf("Move disk %d from rod %c to rod %c\n", n, from_rod, to_rod);
   towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}

int main() {
   int n = 3; // number of disks
   towerOfHanoi(n, 'A', 'C', 'B');
   getch();
   return 0;
}
```

OUTPUT -:



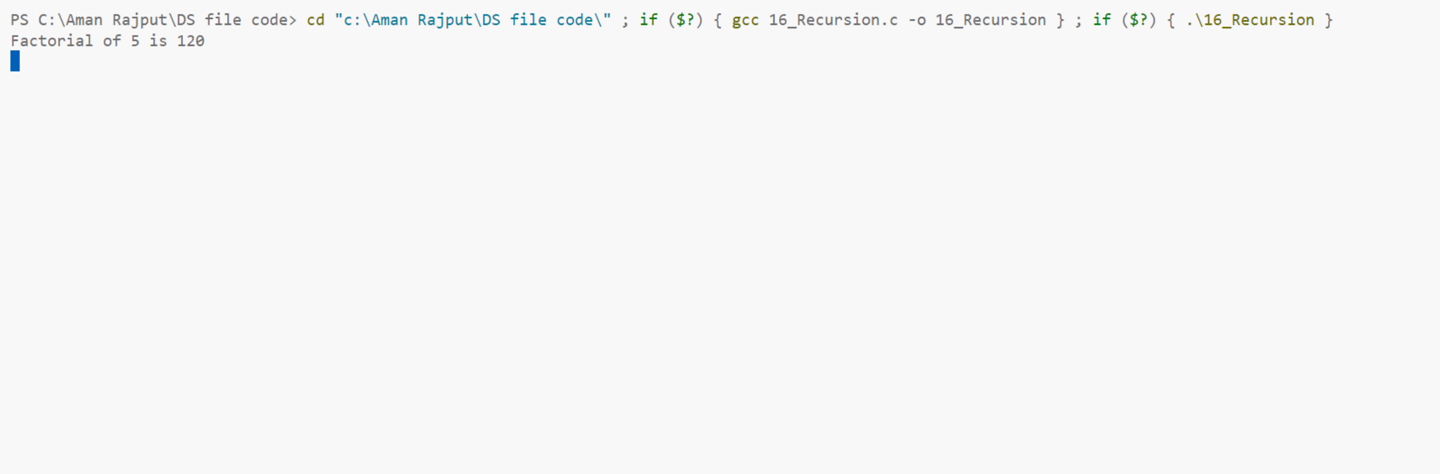
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 15_Tower_Hanoi.c -o 15_Tower_Hanoi } ; if ($?) { .\15_Tower_Hanoi }
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
```

EXPERIMENT NO-:16

Write a C program using RECURSION

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

int factorial(int n) {
   if(n == 0) {
      return 1;
   }
   else {
      return n * factorial(n-1);
   }
}

int main() {
   int n = 5;
   printf("Factorial of %d is %d\n", n, factorial(n));
   getch();
   return 0;
}
```

OUTPUT -:



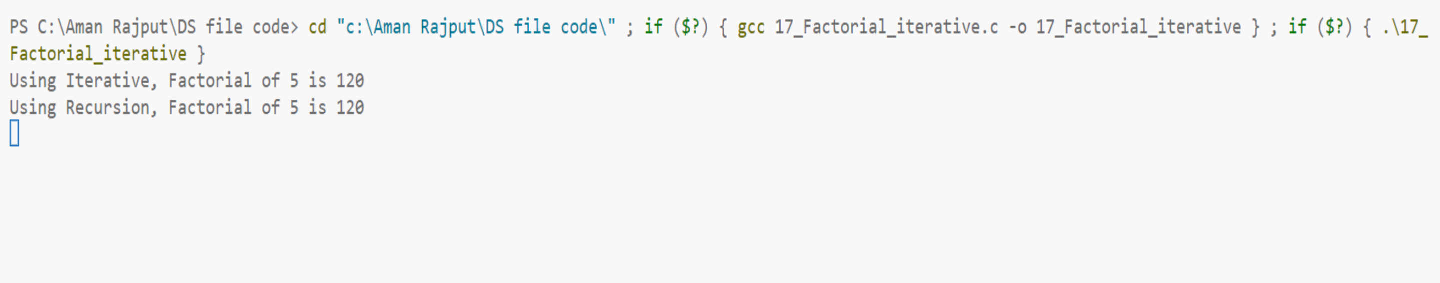
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 16_Recursion.c -o 16_Recursion } ; if ($?) { .\16_Recursion }
Factorial of 5 is 120
```

EXPERIMENT NO-:17

Write a C program to perform FACTORIAL using ITERATIVE & RECURSION METHOD

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

int factorial_ite(int n) {
   int i,result = 1;
   for(i = 1; i <= n; i++) {
      result *= i;
   }
   return result;
}
int factorial_rec(int n) {
   if(n == 0) {
      return 1;
   }
   else {
      return n * factorial_rec(n-1);
   }
}

int main() {
   int n = 5;
   printf("Using Iterative, Factorial of %d is %d\n", n, factorial_ite(n));
   printf("Using Recursion, Factorial of %d is %d\n", n, factorial_rec(n));
   getch();
   return 0;
}
```

OUTPUT -:



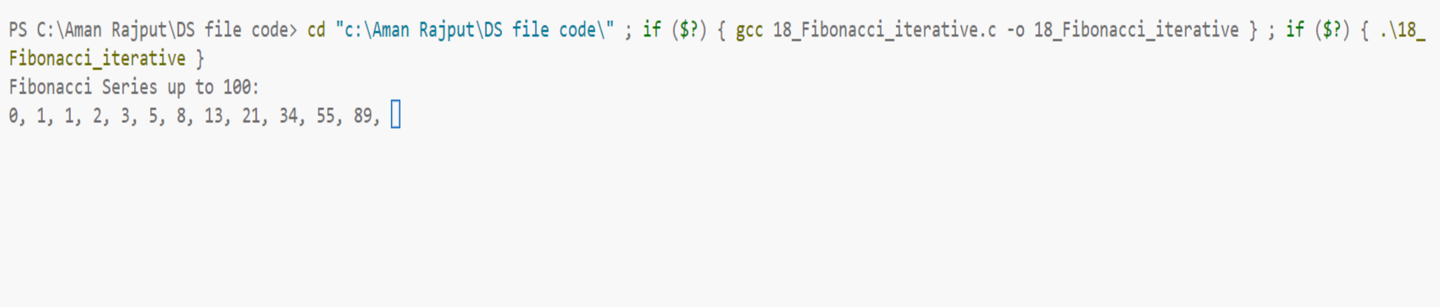
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 17_Factorial_iterative.c -o 17_Factorial_iterative } ; if ($?) { .\17_
Factorial_iterative }
Using Iterative, Factorial of 5 is 120
Using Recursion, Factorial of 5 is 120
```

EXPERIMENT NO-:18

Write a C program to perform FIBONACCI SERIES using the ITERATIVE method

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

void fibonacci(int limit) {
   int num1 = 0, num2 = 1, nextTerm;
   printf("Fibonacci Series up to %d:\n", limit);
   printf("%d, %d, ", num1, num2);
   nextTerm = num1 + num2;
   while(nextTerm <= limit) {
      printf("%d, ", nextTerm);
      num1 = num2;
      num2 = nextTerm;
      nextTerm = num1 + num2;
   }
}

int main() {
   int limit = 100;
   fibonacci(limit);
   getch();
   return 0;
}
```

OUTPUT -:



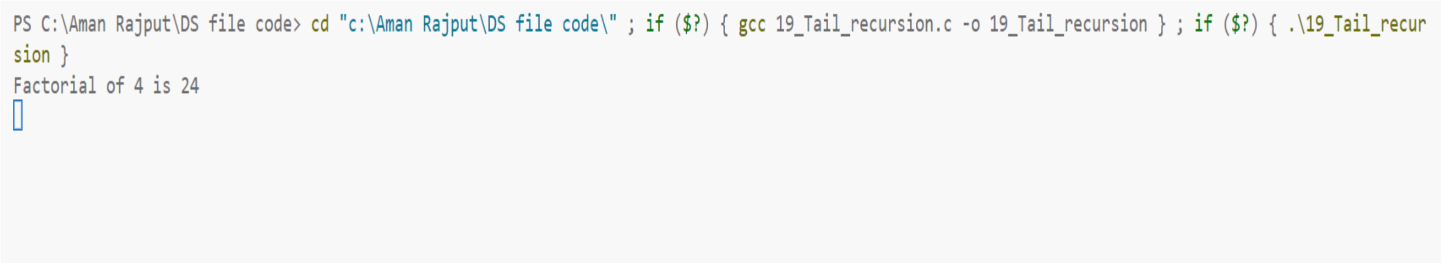
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 18_Fibonacci_iterative.c -o 18_Fibonacci_iterative } ; if ($?) { .\18_
Fibonacci_iterative }
Fibonacci Series up to 100:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 
```

EXPERIMENT NO-:19

Write a C program to perform TAIL RECURSION

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

int tail_factorial(int n, int result) {
    if(n == 0) {
        return result;
    } else {
        return tail_factorial(n - 1, n * result);
    }
}

int main() {
    int n = 4;
    printf("Factorial of %d is %d\n", n, tail_factorial(n, 1));
    getch();
    return 0;
}
```

OUTPUT -:



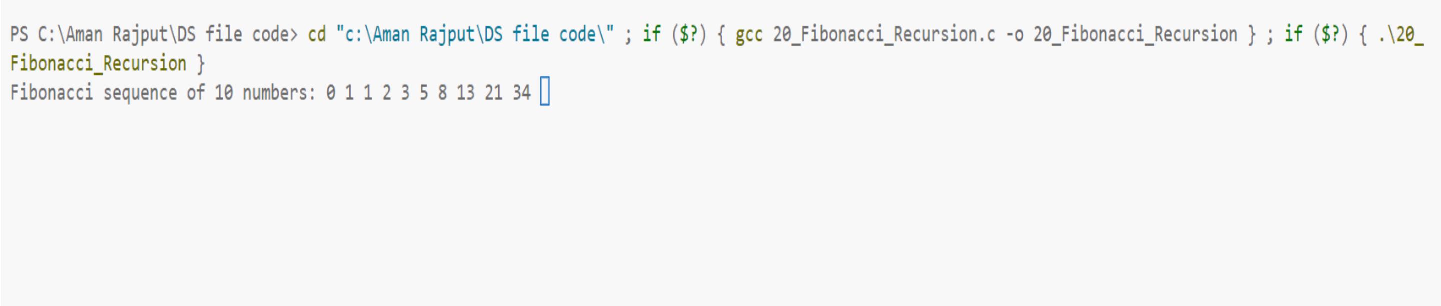
```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 19_Tail_recursion.c -o 19_Tail_recursion } ; if ($?) { .\19_Tail_recur
sion }
Factorial of 4 is 24
```

EXPERIMENT NO-:20

Write a C program to perform FIBONACCI SERIES using the RECURSION method

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

int fibonacci(int n) {
   if(n == 0 || n == 1) {
      return n;
   } else {
      return fibonacci(n-1) + fibonacci(n-2);
   }
}

int main() {
   int i,n = 10;
   printf("Fibonacci sequence of %d numbers: ", n);
   for(i = 0; i < n; i++) {
      printf("%d ", fibonacci(i));
   }
   getch();
   return 0;
}
```

OUTPUT -:



```
PS C:\Aman Rajput\DS file code> cd "c:\Aman Rajput\DS file code\" ; if ($?) { gcc 20_Fibonacci_Recursion.c -o 20_Fibonacci_Recursion } ; if ($?) { .\20_
Fibonacci_Recursion }
Fibonacci sequence of 10 numbers: 0 1 1 2 3 5 8 13 21 34
```