"DATA STRUCTURE AND ALGORITHM LAB"

A LAB RECORD SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE SUBJECT

"DATA STRUCTURE AND ALGORITHM"

OF

Bachelor of Technology (Computer Science)

Submitted by:

MD MERAJUL HAQUE

B.Tech. (Computer Science) 2nd Year
Roll Number: 22BLCS005HY
Enrollment Number: A191069

Submitted to:

MOHTESHAM PASHA QUADRI

Assistant Professor
Department of Computer Science & Information Technology
Maulana Azad National Urdu University, Hyderabad

Department of Computer Science & Information Technology
Maulana Azad National Urdu University, Hyderabad

Maulana Azad National Urdu University
Gachibowli, Hyderabad, Telangana-500032 (India)
(Accredited with "A+" Grade by NAAC)

Certificate

    This is to certify that the lab record file by **MD MERAJUL HAQUE** bearing Enrollment Number **A191069** submitted in partial fulfillment of the requirements for the subject **"DATA STRUCTURE AND ALGORITHM LAB"** with course code **"BTCS360PCP"** in **Bachelor of Technology** (Computer Science) **3rd Semester** during 2022-23 at the **Department of Computer Science & Information Technology** is a bonafide laboratory work carried out by him under my supervision.

    The results presented in this file have been verified and are found to be satisfactory.

Signature of Internal Examiner   **Signature of External Examiner**

INDEX / انڈیکس

LAB PRACTICAL RECORD / لیب عملی ریکارڈ

Program No (پروگرام) -: 01

Aim (مقصد) :- IMPLEMENT THE ARRAY OPERATION (INSERTION)

Algorithm (الگورتھم/طریقہ کار) :-

INSERTION AT THE END OF ARRAY:

- **Step 1:** if UB=MAX, then array is overflow

- **Step 2: Read DATA**
- **Step 3: UB=UB+1**
- **Step 4: arr[UB]=DATA**

INSERTION AT THE BEGINNING OF ARRAY:

- **Step 1: if UB=MAX,then array is overflow Step 2: Read DATA**
- **Step 3: k=UB**
- **Step 4: Repeat step 5 while k>=LB Step 5: arr[K+1]=arr[k]**
- **K=k-1**
- **Step 6: arr[LB]=DATA**
- **Step 7: Stop**

INSERTION AT THE GIVEN POSITION/LOCATION OF ARRAY:

- **Step 1: if UB=MAX,then array is overflow**

- **Step 2: Read DATA and LOCATION**

- **Step 3: k=UB**

- **Step 4: Repeat step 5 while k>=LOCATION Step 5: arr[K+1]=arr[k]**

- K=k-1

- Step 6: arr[LOCATION]=DATA

- Step 7: Stop

Code(کوڈ) :-

```c
#include<stdio.h>
#include <stdlib.h>
#define max 100

void insertion();
void insertionEnd();
void insertionBegin();
void insertionLoc();
void display();

struct array{
   int ub;
   int lb;
   int a[max];
   int size;
}a;
void main(){
   int choice;
   a.lb = -1;
   a.ub = -1;
   printf("Enter the size of Array : ");
   scanf("%d",&a.size);
   insertion();
}
void insertion(){
   int choice,d;
   do{
      printf("\n1. Insert at the End\n2. Insert at the Begin\n3. Insert
at the Location\n4. Display\n5. Exit\n ");
      scanf("%d",&choice);
      switch (choice){
         case 1:
            insertionEnd();
            display();
            break;
         case 2:
            insertionBegin();
```

```c
                    display();
                    break;
                case 3:
                    insertionLoc();
                    display();
                    break;
                case 4:
                    display();
                    break;
                case 5:
                    exit(0);
                default:
                    printf("Wrong choice!!!");
                    break;
        }
    }while(choice!=5);
}
void insertionEnd(){
    int data;
    if(a.ub >= a.size){
        printf("Array is Overflow!!");
    }else{
        printf("Enter element to be inserted : ");
        scanf("%d",&data);
        a.ub++;
        a.a[a.ub] = data;
    }
}
void insertionBegin(){
    int data,i;
    if(a.ub >= a.size){
        printf("Array is Overflow!!");
    }else{
        printf("Enter element to be inserted : ");
        scanf("%d",&data);
        a.ub++;
        for(i=0;i<=a.ub;i++){
            a.a[i+1] = a.a[i];
        }
        a.a[0] = data;
    }
}
void insertionLoc(){
    int data,loc,i;
```

```c
    if(a.ub >= a.size){
        printf("Array is Overflow!!");
    }else{
        printf("Enter element to be inserted : ");
        scanf("%d",&data);
        printf("Enter position : ");
        scanf("%d",&loc);
        a.ub++;
        for(i=loc;i<=a.ub;i++){
            a.a[i+1] = a.a[i];
        }
        a.a[loc] = data;
    }
}
void display(){
    int i;
    if(a.ub == -1){
        printf("No Elements!!");
    }else{
        printf("Array Elements : ");
        for(i=0;i<=a.ub;i++){
            printf("%d\t",a.a[i]);
        }
    }
}
```

Output(نتیجہ):-

```
D:\B.tech\DSA by LB\DSA-Co    ×    +    ∨

Enter the size of Array : 4

1. Insert at the End
2. Insert at the Begin
3. Insert at the Location
4. Display
5. Exit
 1
Enter element to be inserted : 10
Array Elements : 10
1. Insert at the End
2. Insert at the Begin
3. Insert at the Location
4. Display
5. Exit
 2
Enter element to be inserted : 20
Array Elements : 20      10
1. Insert at the End
2. Insert at the Begin
3. Insert at the Location
4. Display
5. Exit
 3
Enter element to be inserted : 302
Enter position : 2
Array Elements : 20      10      302
1. Insert at the End
2. Insert at the Begin
3. Insert at the Location
4. Display
5. Exit
 4
Array Elements : 20      10      302
1. Insert at the End
2. Insert at the Begin
3. Insert at the Location
4. Display
5. Exit
5


--------------------------------
Process exited after 347.4 seconds with return value 0
```

Program No (پروگرام) :- 02

Aim (مقصد) :- IMPLEMENT THE ARRAY OPERATION (DELETION)

Algorithm (الگورتھم/طریقہ کار) :-

BEGINNING:
- Step 1: if UB==0, the array is underflow
- Step 2: k=LB
- Step 3: Repeat the step 4 while k<UB Step 4: arr[k]=arr[k+1]
- K=k+1
- Step 5: arr[UB]=NULL
- UB+UB-1
- Step 6: stop

DELETION AT END:
- Step 1: if UB==0 then array is underflow
- Step 2: arr[UB]=NULL
- UB=UB-1
- Step 3: stop

DELETION AT THE GIVEN LOCATION:
- Step 1: if UB==0, the array is underflow
- Step 2: Read LOCATION
- K=LOCATION
- Step 3: Repeat the step 4 while k<UB Step 4: arr[k]=arr[k+1]
- K=k+1
- Step 5: arr[UB]=NULL
- UB+UB-1
- Step 6: stop

Code(کوڈ) :-

```c
#include<stdio.h>
#include <stdlib.h>
#define max 100

void deletion();
void deletionEnd();
void deletionBegin();
void deletionLoc();
void display();

struct array{
    int ub;
    int lb;
    int a[max];
    int size;
}a;
void main(){
    int choice,i;
    a.lb = -1;
    printf("Enter size of Array : ");
    scanf("%d",&a.size);
    printf("Enter Elements : \n");
    for(i=0;i<a.size;i++){
        scanf("%d",&a.a[i]);
    }
    a.ub = a.size-1;
    display();
    deletion();
}
void deletion(){
```

```c
    int choice,d;
    do{
        printf("\n1. Delete at the End\n2. Delete at the Begin\n3. Delete
at the Location\n4. Display\n5. Exit\n ");
        scanf("%d",&choice);
        switch (choice){
            case 1:
                deletionEnd();
                display();
                break;
            case 2:
                deletionBegin();
                display();
                break;
            case 3:
                deletionLoc();
                display();
                break;
            case 4:
                display();
                break;
            case 5:
                exit(0);
            default:
                printf("Wrong choice!!!");
                break;
        }
    }while(choice!=5);
}
void deletionEnd(){
    int data;
    if(a.ub == a.lb){
```

```
        printf("Array is Underflow!!");
    }else{
        printf("Deleted Element : %d\n",a.a[a.ub]);
        a.ub--;
    }
}
void deletionBegin(){
    int data,i;
    if(a.ub == a.lb){
        printf("Array is Underflow!!");
    }else{
        printf("Deleted Element : %d\n",a.a[0]);
        for(i=0;i<a.ub;i++){
            a.a[i] = a.a[i+1];
        }
        a.ub--;
    }
}
void deletionLoc(){
    int data,loc,i;
    if(a.ub == a.lb){
        printf("Array is Underflow!!");
    }else{
        printf("Enter position : ");
        scanf("%d",&loc);
        printf("Deleted Element : %d\n",a.a[loc-1]);

        for(i=loc-1;i<a.ub;i++){
            a.a[i] = a.a[i+1];
        }
        a.ub--;
    }
```

```
}
void display(){
    int i;
    if(a.ub == -1){
        printf("No Elements!!");
    }else{
        printf("Array Elements : ");
        for(i=0;i<=a.ub;i++){
            printf("%d\t",a.a[i]);
        }
    }
}
```

Output(نتیجہ):-

```
              D:\B.tech\DSA by LB\DSA-Co    ×    +    ∨

Enter size of Array : 4
Enter Elements :
10
20
30
40
Array Elements : 10      20      30      40
1. Delete at the End
2. Delete at the Begin
3. Delete at the Location
4. Display
5. Exit
 3
Enter position : 3
Deleted Element : 30
Array Elements : 10      20      40
1. Delete at the End
2. Delete at the Begin
3. Delete at the Location
4. Display
5. Exit
 1
Deleted Element : 40
Array Elements : 10      20
1. Delete at the End
2. Delete at the Begin
3. Delete at the Location
4. Display
5. Exit
 2
Deleted Element : 10
Array Elements : 20
1. Delete at the End
2. Delete at the Begin
3. Delete at the Location
4. Display
5. Exit
 4
Array Elements : 20
1. Delete at the End
2. Delete at the Begin
3. Delete at the Location
4. Display
5. Exit
 5


--------------------------------
Process exited after 40.08 seconds with return value 0
Press any key to continue . . . |
```

Program No (پروگرام) :- 03

Aim (مقصد) :- Implementation of Stack Operation (Push, Pop and Peak)

Algorithm (الگوردتھم/طریقہ کار) :-

Push Operation

- The process of putting a new data element onto stack is known as a Push

  Operation. Push operation involves a series of steps –

- Step 1 – Checks if the stack is full.

- Step 2 – If the stack is full, produces an error and exit.

- Step 3 – If the stack is not full, increments top to point next empty

  space. Step 4 – Adds data element to the stack location, where top

  is pointing. Step 5 – Returns success.

POP OPERATION

- Step 1 – Checks if the stack is empty.
- Step 2 – If the stack is empty, produces an error and exit.
- Step 3 – If the stack is not empty, accesses the data element at which top
  is pointing.
- Step 4 – Decreases the value of top by 1.
- Step 5 – Returns success.

PEAK OPERATION

- Step 1:begin procedure peek

- Step 2: return stack[top]

- Step 3: end procedure

Code(کوڈ) -:

```c
#include<stdio.h>
#define max 100
void push();
void pop();
void display();
void peek();
struct stack{
    int top;
    int arr[max];
}s;
void main(){
    int x;
    s.top = -1;
    do{
        printf("\n1. Push\n2. Pop\n3. Display\n4. Peek\n5. Exit\n");
        printf("Choose Option : ");
        scanf("%d",&x);
        switch(x){
            case 1:
                push();
                display();
                break;
            case 2:
                pop();
                display();
                break;
            case 3:
                display();
                break;
            case 4:
```

```c
            peek();
            break;
        case 5:
            break;
    }
  }while(x!=5);
}
void push(){
   int data;
   if(s.top == max-1){
      printf("Stack Overflow!!");
   }
   else{
      printf("Enter the element to be pushed : ");
      scanf("%d",&data);
      s.top++;
      s.arr[s.top] = data;
   }
}
void pop(){
   int data;
   if(s.top == -1){
      printf("Stack Underflow!!\n");
   }
   else{
      data = s.arr[s.top];
      s.top--;
      printf("Popped Element is %d\n",data);
   }
}
void display(){
   int i=0;
```

```c
   if(s.top == -1){
      printf("Stack Underflow!!");
   }else{
      printf("Elements are : ");
      for(i=0;i<=s.top;i++){
         printf("%d\t",s.arr[i]);
      }
   }
}
void peek(){
   if(s.top == -1){
      printf("Stack Underflow!!");
   }else{
      printf("Peek Element is %d",s.arr[s.top]);
   }
}
```

Output(نتیجہ):-

```
[C:\]  D:\B.tech\DSA by LB\DSA-Cor    ×      +    ∨

        1. Push
        2. Pop
        3. Display
        4. Peek
        5. Exit
        Choose Option : 1
        Enter the element to be pushed : 20
        Elements are : 20
        1. Push
        2. Pop
        3. Display
        4. Peek
        5. Exit
        Choose Option : 1
        Enter the element to be pushed : 30
        Elements are : 20        30
        1. Push
        2. Pop
        3. Display
        4. Peek
        5. Exit
        Choose Option : 1
        Enter the element to be pushed : 10
        Elements are : 20        30        10
        1. Push
        2. Pop
        3. Display
        4. Peek
        5. Exit
        Choose Option : 2
        Popped Element is 10
        Elements are : 20        30
        1. Push
        2. Pop
        3. Display
        4. Peek
        5. Exit
        Choose Option : 3
        Elements are : 20        30
        1. Push
        2. Pop
        3. Display
        4. Peek
        5. Exit
        Choose Option : 4
        Peek Element is 30
        1. Push
        2. Pop
        3. Display
        4. Peek
        5. Exit
        Choose Option : 5


        ---------------------------------
        Process exited after 46.51 seconds with return value 5
        Press any key to continue . . . |
```

Program No (پروگرام) :- 0

Aim (مقصد) :- Implementation of simple queue operation(Enqueue and Dequeue)

Algorithm (الگورتھم/طریقہ کار) :-

Queue operations work as follows :

- two pointers FRONT and REAR
- FRONT track the first element of the queue
- REAR track the last element of the queue
- initially, set value of FRONT and REAR to -1

Enqueue Operation :

- Step 1: check if the queue is full
- Step 2: for the first element, set the value of FRONT to 0
- Step 3: increase the REAR index by 1
- Step 4: add the new element in the position pointed to by REAR

Dequeue Operation :

- Step 1: check if the queue is empty
- Step 2: return the value pointed by FRONT
- Step 3: increase the FRONT index by 1
- Step 4: for the last element, reset the values of FRONT and REAR to -1

Code(کوڈ) :-

```
#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#define max 100


//declaring function globally

void enQueue();

void deQueue();
```

```c
void display();
int Queue[max], front = -1, rear = -1;
int i,value;


int main(){
    int choice;
    do{
        printf("\n1. EnQueue \n2. DeQueue \n3. Display\n4. Exit\nEnter
Your Choice : ");
    scanf("%d",&choice);


        switch (choice)
        {
        case 1:
            enQueue();
            display();
            break;
        case 2:
            deQueue();
            display();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
        default:
            printf("Wrong selection ! \n");
            break;
        }
    }while(choice!=4);
    return 0;
```

```c
}
void enQueue(){
    int data;
    if(rear == max-1){
        printf("Queue is full!!");
    }else{
        printf("Enter data to enQueue : ");
        scanf("%d",&data);
        if(front == -1){
            front = 0;
        }
        rear++;
        Queue[rear] = data;
    }
}
void deQueue(){
    if(front == -1){
        printf("Queue is empty !!\n");
    }else{
        printf("Deleted : %d\n",Queue[front]);
        front++;
        if(front>rear){
            front = rear = -1;
        }
    }
}
void display(){
    if(rear == -1){
        printf("Queue is Empty !!! \n");
    }else{
        int i;
        printf("Queue elements : ");
```

```c
        for(i = front; i<= rear; i++){
            printf("%d\t", Queue[i]);
        }
    }
}
```

Output(نتیجہ):-

```
 c:\  D:\B.tech\DSA by LB\DSA-Co    ×    +    ∨

    1. EnQueue
    2. DeQueue
    3. Display
    4. Exit
    Enter Your Choice : 1
    Enter data to enQueue : 100
    Queue elements : 100
    1. EnQueue
    2. DeQueue
    3. Display
    4. Exit
    Enter Your Choice : 1
    Enter data to enQueue : 30
    Queue elements : 100    30
    1. EnQueue
    2. DeQueue
    3. Display
    4. Exit
    Enter Your Choice : 3
    Queue elements : 100    30
    1. EnQueue
    2. DeQueue
    3. Display
    4. Exit
    Enter Your Choice : 2
    Deleted : 100
    Queue elements : 30
    1. EnQueue
    2. DeQueue
    3. Display
    4. Exit
    Enter Your Choice : 4


    --------------------------------
    Process exited after 33.19 seconds with return value 0
    Press any key to continue . . . |
```

Program No (پروگرام) :- 0

Aim (مقصد) :-

C program for implementation of Bubble sort

Algorithm (الگورتھم / طریقہ کار) :-

1. begin BubbleSort(arr)
2.   for all array elements
3.   if arr[i] > arr[i+1]
4.   swap(arr[i], arr[i+1])
5.   end if
6.   end for
7.   return arr
8. end BubbleSort

Code(کوڈ) :-

```c
// C program for implementation of Bubble sort
#include<stdio.h>
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void bubbleSort(int arr[], int n)
{
int i, j;
for (i = 0; i<n-1; i++)

    for (j = 0; j<n-i-1; j++)
        if (arr[j] >arr[j+1])
            swap(&arr[j], &arr[j+1]);
}
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i<size; i++)
        printf("%d ", arr[i]);
```

```c
    printf("\n");
}
int main()
{
    int arr[100], n,i;
    printf("Enter the no. of element you want to Sort. \n");
    scanf("%d", &n);
    printf("Now Enter the Element : \n");
    for(i = 0; i<n; i++){
        scanf("%d", &arr[i]);
    }
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Output(نتیجہ):-

```
[C:\] D:\B.tech\DSA by LB\DSA-Co    ×    +    ∨

Enter the no. of element you want to Sort.
5
Now Enter the Element :
20
10
40
15
35
Sorted array:
10 15 20 35 40

--------------------------------
Process exited after 25.25 seconds with return value 0
Press any key to continue . . .
```

Program No (پروگرام) :- 01

Aim (مقصد) :- Implementation of selection sort

Algorithm (الگورتھم/طریقہ کار) :-

- Step 1 - If the element is the first element, assume that it is already sorted. Return 1.
- Step2 - Pick the next element, and store it separately in a key.
- Step3 - Now, compare the key with all elements in the sorted array.
- Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
- Step 5 - Insert the value.
- Step 6 - Repeat until the array is sorted.

Code(کوڈ) :-

```c
// C program for implementation of selection sort
#include<stdio.h>
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}


void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i<n-1; i++)
    {
        // Find the minimum element in unsorted array
```

```c
        min_idx = i;
        for (j = i+1; j<n; j++)
          if (arr[j] <arr[min_idx])
            min_idx = j;


        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}


/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i<size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main()
{
    int arr[100], n,i;
    printf("Enter the no. of element you want to Sort. \n");
    scanf("%d", &n);
    printf("Now Enter the Element : \n");
    for(i = 0; i<n; i++){
        scanf("%d", &arr[i]);
    }
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Output(نتیجہ):-

```
⌑  D:\B.tech\DSA by LB\DSA-Co   ✕    +   ∨

Enter the no. of element you want to Sort.
5
Now Enter the Element :
30
35
10
2
15
Sorted array:
2 10 15 30 35

--------------------------------
Process exited after 21.91 seconds with return value 0
Press any key to continue . . .
```

Program No (پروگرام) :- 01

Aim (مقصد) :- SORTING TECHNIQUE (INSERTION SORT)

Algorithm (الگوردتھم/طریقہ کار) :-

- Step 1 - If the element is the first element, assume that it is already sorted. Return 1.
- Step2 - Pick the next element, and store it separately in a key. Step3 - Now, compare the key with all elements in the sorted array.
- Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
- Step 5 - Insert the value.
- Step 6 - Repeat until the array is sorted.

Code(کوڈ) :-

```c
// C program for insertion sort
#include<math.h>
#include<stdio.h>

void insertionSort(int arr[], int n)
{
    int i, key, j;
    for (i = 1; i<n; i++)
    {
        key = arr[i];
        j = i - 1;
        while (j>= 0 && arr[j]>key){
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
```

```c
}


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


int main()
{
    int arr[100], n,i;
    printf("Enter the number of element you want to Sort : ");
    scanf("%d", &n);
    printf("Now Enter the Element : \n");
    for(i = 0; i<n; i++){
        scanf("%d", &arr[i]);
    }
    insertionSort(arr, n);
    display(arr, n);
    return 0;
}
```

Output(نتیجہ):-

```
C:\ D:\B.tech\DSA by LB\DSA-Co    ×    +    ∨

Enter the number of element you want to Sort : 5
Now Enter the Element :
12
23
10
15
20
Sorted array: 10          12        15        20        23
--------------------------------
Process exited after 16.35 seconds with return value 0
Press any key to continue . . . |
```

Output(نتیجہ):-

Program No (پروگرام) :- 01

Aim (مقصد) :-

Algorithm (الگورتھم/طریقہ کار) :-

Code(کوڈ) :-

Output(نتیجہ):-

Program No (پروگرام) :- 01