**“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**

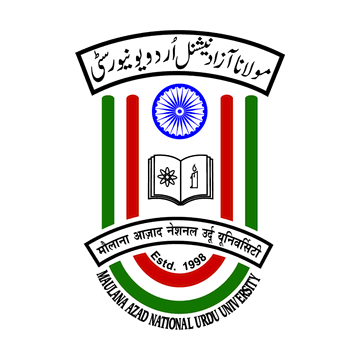
**Submitted to:**

**MOHTESHAM PASHA QUADRI**

**Assistant Professor**

**Department of Computer Science & Information Technology**

**Maulana Azad National Urdu University, Hyderabad**

****

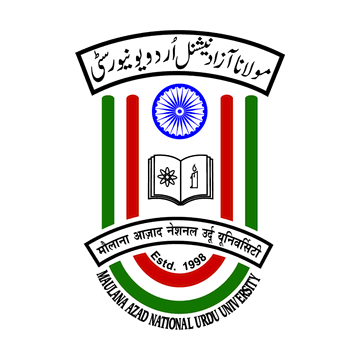
**Department of Computer Science & Information Technology**

**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 ریکار**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **NAME OF EXPERIMRNT** | **PAGE NUMBER** |
| **1** | **Implement the Array Operation ( Insertion )** |  |
| **2** | **Implement the Array Operation ( Deletion )** |  |
| **3** | **Implementation of Stack Operation ( Push, Pop and Peak )** |  |
| **4** | **Implementation of Infix to Postfix** |  |
| **5** | **Implementation of Simple Queue Operation ( Enqueue and Dequeue )** |  |
| **6** | **Implementation of Circular Queue Operation ( Enqueue and Dequeue )** |  |
| **7** | **Sorting Technique ( Bubble Sort )** |  |
| **8** | **Sorting Technique ( Selection Sort )** |  |
| **9** | **Sorting Technique ( Insertion Sort )** |  |
| **10** | **Sorting Technique ( Quick Sort )** |  |

**) :- 01*پروگرام* No ( Program**

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

**) :-*الگورتھم/طریقہ کار*Algorithm (**

### 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*) :-**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct array{**

**int \*arr;**

**int size;**

**int ub;**

**int lb;**

**};**

**int isFull(struct array \*a){**

**if(a->ub==a->size){**

**printf("The array is overflow\n");**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**int isEmpty(struct array \*a){**

**if(a->ub==a->lb){**

**printf("The array is underflow\n");**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**void insertion\_at\_beginning(struct array \*a){**

**if(isFull(a)){**

**printf("\nArray is overflow so we cannot insert data\n");**

**}**

**else{**

**int data;**

**printf("Enter the data which you want to insert:");**

**scanf("%d",&data);**

**a->ub++;**

**for(int i=a->ub;i>0;i--){**

**a->arr[i]=a->arr[i-1];**

**}**

**a->arr[0]=data;**

**}**

**}**

**void insertion\_at\_end(struct array \*a){**

**if(isFull(a)){**

**printf("Array is overflow so we cannot insert data");**

**}**

**else{**

**int data;**

**printf("Enter the data which you want to insert:");**

**scanf("%d",&data);**

**a->ub++;**

**a->arr[a->ub]=data;**

**}**

**}**

**void insertion\_at\_position(struct array \*a){**

**if(isFull(a)){**

**printf("Array is overflow so we cannot insert data");**

**}**

**else{**

**int data;**

**printf("Enter the data which you want to insert:");**

**scanf("%d",&data);**

**printf("enter the position which you want to insert the data:");**

**int position;**

**scanf("%d",&position);**

**a->ub++;**

**for(int i=a->ub;i>position;i--){**

**a->arr[i]=a->arr[i-1];**

**}**

**a->arr[position]=data;**

**}**

**}**

**void display(struct array \*a){**

**printf("\nThe updated array is: ");**

**for(int i=0;i<=a->ub;i++){**

**printf("%d ",a->arr[i]);**

**}**

**}**

**int main(){**

**struct array \*a;**

**a->ub=-1;**

**a->lb=-1;**

**int choice;**

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

**scanf("%d",&a->size);**

**a->arr=(int \*)malloc(sizeof(int)\*a->size);**

**do{**

**printf("\n1.Insertion at beginning\n2.Insertion at end\n3.Insertion at any index\n4.Display the array\n5.Exit\n");**

**printf("\nEnter the option:");**

**scanf("%d",&choice);**

**switch(choice){**

**case 1:**

**insertion\_at\_beginning(a); break;**

**case 2:**

**insertion\_at\_end(a); break;**

**case 3:**

**insertion\_at\_position(a); break;**

**case 4:**

**display(a); break;**

**case 5:**

**printf("\n!!Exit the program!!\n"); break;**

**default:**

**printf("\n!!Please enter the right option!!\n");**

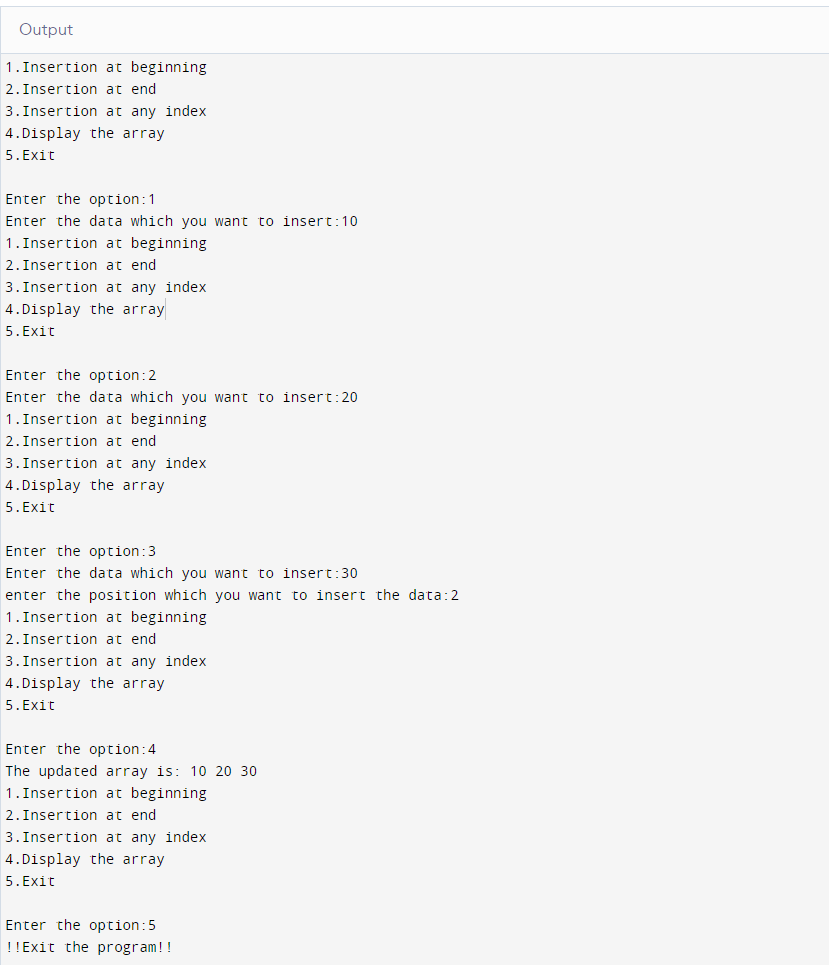
**break;**

**}**

**}while(choice!=5);**

**}**

**Output(*نتیجہ*):-**



**) :- 02*پروگرام* No ( Program**

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

**) :-*الگورتھم/طریقہ کار*Algorithm (**

**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*) :-**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct array{**

**int \*arr;**

**int size;**

**int ub;**

**int lb;**

**};**

**int isFull(struct array \*a){**

**if(a->ub==a->size){**

**printf("Array is overflow\n");**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**int isEmpty(struct array \*a){**

**if(a->ub==a->lb){**

**printf("Array is underflow\n");**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**void delete\_at\_beginning(struct array \*a){**

**if(isEmpty(a)){**

**printf("\nArray is underflow so we cannot delete the array element");**

**}**

**else{**

**printf("\nDeleted element is: %d",a->arr[0]);**

**for(int i=0;i<a->ub;i++){**

**a->arr[i]=a->arr[i+1];**

**}**

**// a->arr[a->ub]=NULL; a->ub--;**

**}**

**}**

**void delete\_at\_end(struct array \*a){**

**if(isEmpty(a)){**

**printf("\nThe array is underflow so we cannot delete the array element");**

**}**

**else{**

**printf("\nThe deleted element is: %d",a->arr[a->ub]);**

**// a->arr[a->ub]=NULL; a->ub--;**

**}**

**}**

**void delete\_at\_position(struct array \*a){**

**if(isEmpty(a)){**

**printf("\nThe array is underflow so we cannot delete the array element");**

**}**

**else{**

**printf("Enter the position which you want to delete the element:");**

**int position;**

**scanf("%d",&position);**

**printf("\nDeleted element is: %d",a->arr[position-1]);**

**for(int i=position-1;i<a->ub;i++){**

**a->arr[i]=a->arr[i+1];**

**}**

**a->ub--;**

**}**

**}**

**void display(struct array \*a){**

**printf("\nUpdated array is: ");**

**for(int i=0;i<=a->ub;i++){**

**printf("%d ",a->arr[i]);**

**}**

**}**

**int main(){**

**struct array \*a;**

**a->ub=-1;**

**a->lb=-1;**

**int choice;**

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

**scanf("%d",&a->size);**

**a->arr=(int \*)malloc(sizeof(int)\*a->size);**

**a->ub++;**

**a->arr[a->ub]=12;**

**a->ub++;**

**a->arr[a->ub]=13;**

**a->ub++;**

**a->arr[a->ub]=16;**

**do{**

**printf("\n1.Deletion at Beginning\n2.Deletion at end\n3.Deletion at any index\n4.Display the array\n5.Exit\n");**

**printf("\nenter the option:");**

**scanf("%d",&choice);**

**switch(choice){**

**case 1:**

**delete\_at\_beginning(a);**

**break;**

**case 2:**

**delete\_at\_end(a);**

**break;**

**case 3:**

**delete\_at\_position(a);**

**break;**

**case 4:**

**display(a);**

**break;**

**case 5:**

**printf("\n!!Exit the program!!\n");**

**break;**

**default:**

**printf("\n!!please enter the right option!!\n");**

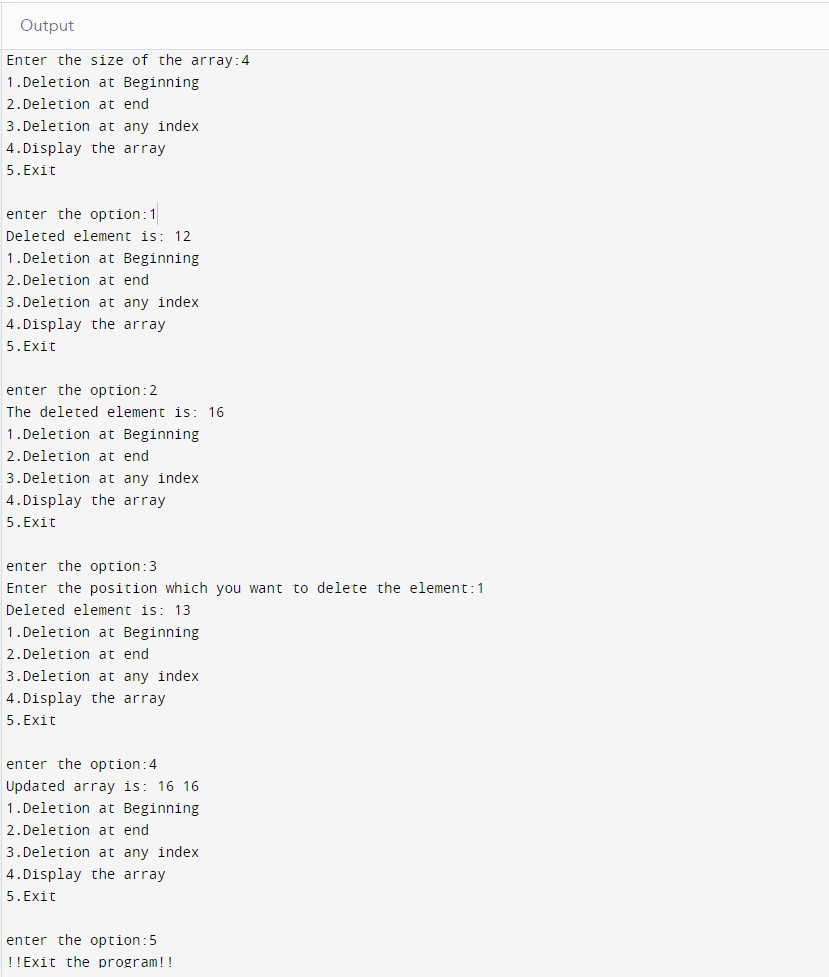
**break;**

**}**

**}while(choice!=5);**

**}**

**Output(*نتیجہ*):-**



**) :- 01*پروگرام* No ( Program**

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

**) :-*الگورتھم/طریقہ کار*Algorithm (**

**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*) :-**

**Output(*نتیجہ*):-**

**) :- 01*پروگرام* No ( Program**

#### Aim (*مقصد*) :-

**) :-*الگورتھم/طریقہ کار*Algorithm (**

**Output(*نتیجہ*):-**