<u>CHAUDHARY HAMDAN</u>

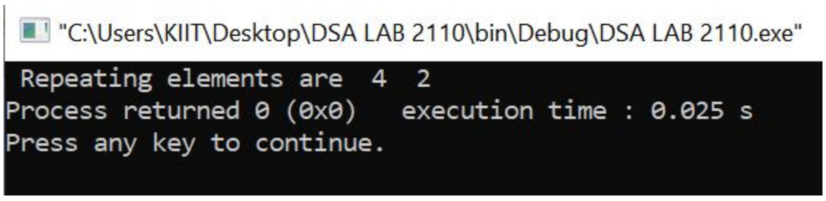
<u>1905387</u>
<u>DSA LAB</u>
<u>Date : 09-11-2020</u>

<u>Faculty in charge : Meghna Ma’am</u>

QUESTION NUMBER-1(ARRAYS)

- **Program to find repeating element in an array (duplicate elements)**

```
#include<stdio.h>

#include<stdlib.h>

void printRepeating(int arr[], int size)

{

    int i, j;

    printf(" Repeating elements are ");

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

        for(j = i+1; j < size; j++)

            if(arr[i] == arr[j])

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

}

int main()

{

    int arr[] = {4, 2, 4, 5, 2, 3, 1};

    int arr_size = sizeof(arr)/sizeof(arr[0]);

    printRepeating(arr, arr_size);

    return 0;

}
```



```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"
 Repeating elements are  4  2
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.
```

- **Program to remove duplicate elements in an array**

Program to remove duplicate elements in an array (sorted and unsorted array cases) is discussed here. Given an array, all the duplicate elements of the array are removed.

For example, consider the array.

case 1: Remove duplicates from sorted array

Input: arr = {1, 2, 3, 4, 4}

Output: arr = {1, 2, 3, 4}

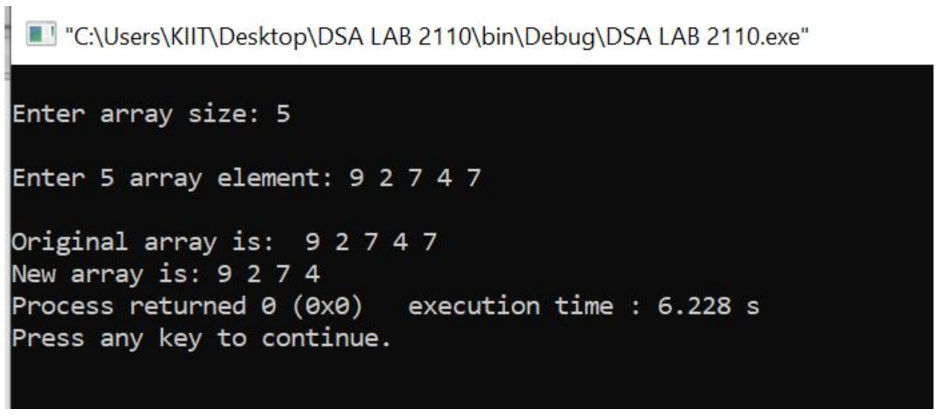
case 2: Remove duplicates from unsorted array

Input: arr = {9, 2, 7, 4, 7}

Output: arr = {9, 2, 7, 4}

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a[20], i, j, k, n;

    printf("\nEnter array size: ");
    scanf("%d", &n);

    printf("\nEnter %d array element: ", n);
    for(i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\nOriginal array is: ");
    for(i = 0; i < n; i++)
    {
        printf(" %d", a[i]);
    }

    printf("\nNew array is: ");
    for(i = 0; i < n; i++)
    {
        for(j = i+1; j < n; )
        {
            if(a[j] == a[i])
            {
                for(k = j; k < n; k++)
                {
                    a[k] = a[k+1];
                }
                n--;
            }
            else
            {
                j++;
            }
        }
    }

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

Output :



```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

Enter array size: 5

Enter 5 array element: 9 2 7 4 7

Original array is:  9 2 7 4 7
New array is: 9 2 7 4
Process returned 0 (0x0)   execution time : 6.228 s
Press any key to continue.
```

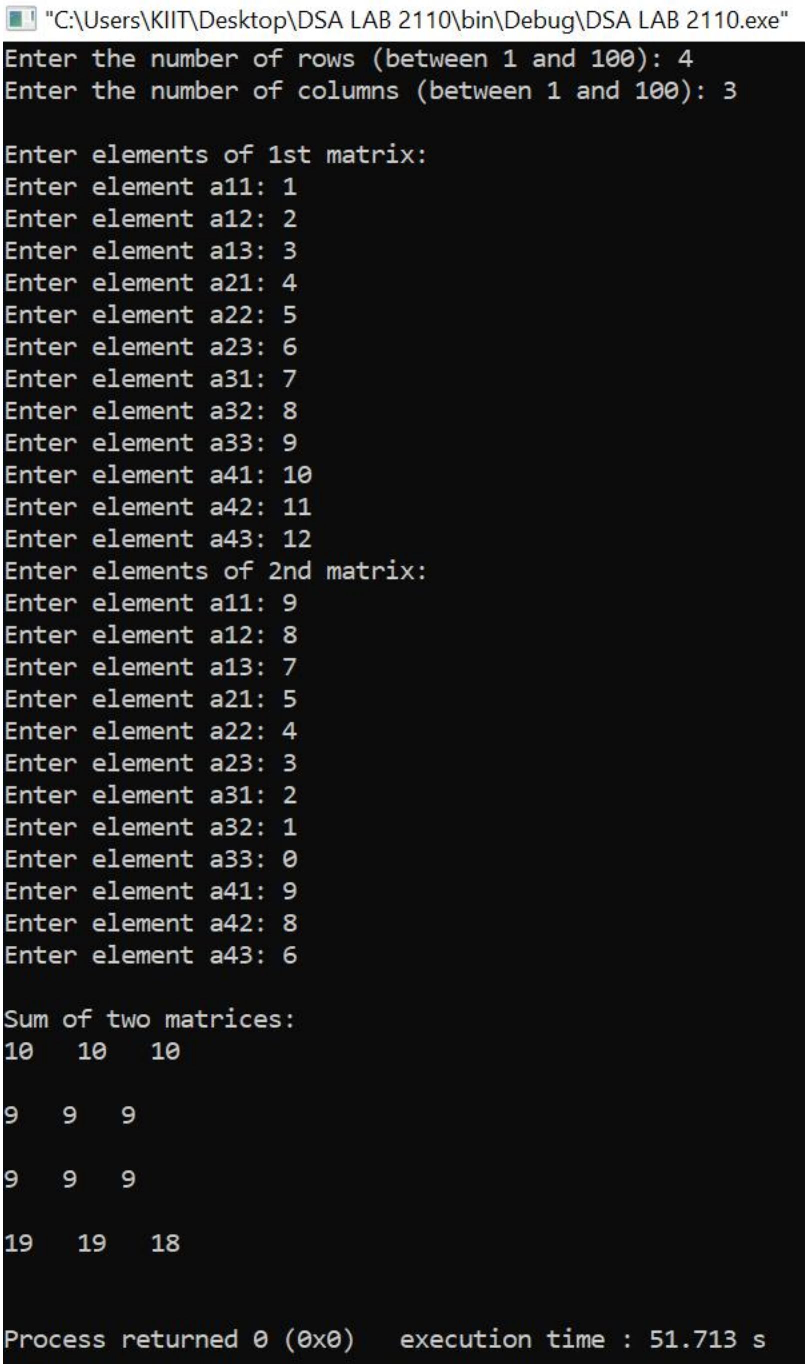
QUESTION NUMBER-2 (2-D ARRAYS)

- **Program to add two matrices**

Program :

```
#include <stdio.h>
int main() {
        int r, c, a[100][100], b[100][100], sum[100][100], i, j;
        printf("Enter the number of rows (between 1 and 100): ");
        scanf("%d", &r);
        printf("Enter the number of columns (between 1 and 100): ");
        scanf("%d", &c);

        printf("\nEnter elements of 1st matrix:\n");
        for (i = 0; i < r; ++i)
                for (j = 0; j < c; ++j) {
                        printf("Enter element a%d%d: ", i + 1, j + 1);
                        scanf("%d", &a[i][j]);
                }

        printf("Enter elements of 2nd matrix:\n");
        for (i = 0; i < r; ++i)
                for (j = 0; j < c; ++j) {
                        printf("Enter element a%d%d: ", i + 1, j + 1);
                        scanf("%d", &b[i][j]);
                }

        for (i = 0; i < r; ++i)
                for (j = 0; j < c; ++j) {
                        sum[i][j] = a[i][j] + b[i][j];
                }

        printf("\nSum of two matrices: \n");
        for (i = 0; i < r; ++i)
                for (j = 0; j < c; ++j) {
                        printf("%d      ", sum[i][j]);
                        if (j == c - 1) {
                                printf("\n\n");
                        }
                }

        return 0;
}
```

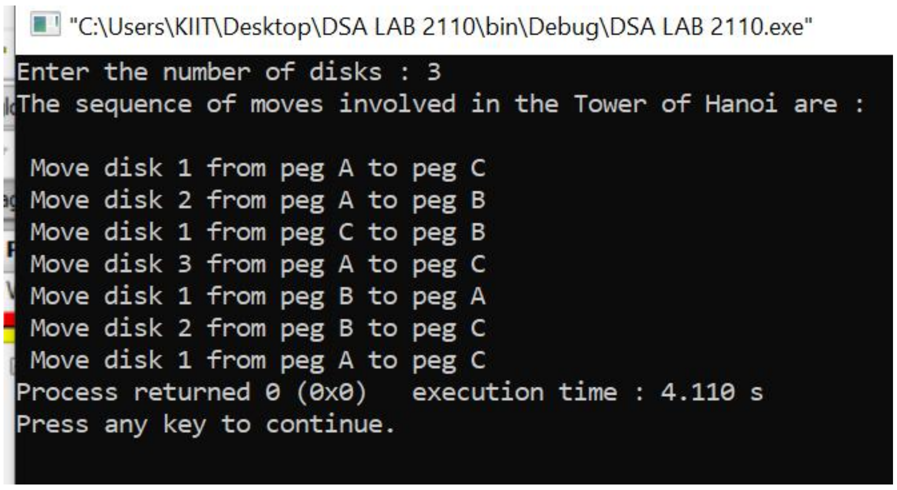
Output :



```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"
Enter the number of rows (between 1 and 100): 4
Enter the number of columns (between 1 and 100): 3

Enter elements of 1st matrix:
Enter element a11: 1
Enter element a12: 2
Enter element a13: 3
Enter element a21: 4
Enter element a22: 5
Enter element a23: 6
Enter element a31: 7
Enter element a32: 8
Enter element a33: 9
Enter element a41: 10
Enter element a42: 11
Enter element a43: 12
Enter elements of 2nd matrix:
Enter element a11: 9
Enter element a12: 8
Enter element a13: 7
Enter element a21: 5
Enter element a22: 4
Enter element a23: 3
Enter element a31: 2
Enter element a32: 1
Enter element a33: 0
Enter element a41: 9
Enter element a42: 8
Enter element a43: 6

Sum of two matrices:
10   10   10

9   9   9

9   9   9

19   19   18


Process returned 0 (0x0)   execution time : 51.713 s
```

❖ (RECURSION)- TOWERS OF HANOI.

```
#include <stdio.h>

void towers(int, char, char, char);

int main()
{
        int num;

        printf("Enter the number of disks : ");
        scanf("%d", &num);
        printf("The sequence of moves involved in the Tower of Hanoi are :\n");
        towers(num, 'A', 'C', 'B');
        return 0;
}
void towers(int num, char from, char to, char temp)
{
        if (num == 1)
        {
                printf("\n Move disk 1 from peg %c to peg %c", from, to);
                return;
        }
        towers(num - 1, from, temp, to);
        printf("\n Move disk %d from peg %c to peg %c", num, from, to);
        towers(num - 1, temp, to, from);
}
```


"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from peg A to peg C
 Move disk 2 from peg A to peg B
 Move disk 1 from peg C to peg B
 Move disk 3 from peg A to peg C
 Move disk 1 from peg B to peg A
 Move disk 2 from peg B to peg C
 Move disk 1 from peg A to peg C
Process returned 0 (0x0)   execution time : 4.110 s
Press any key to continue.
```

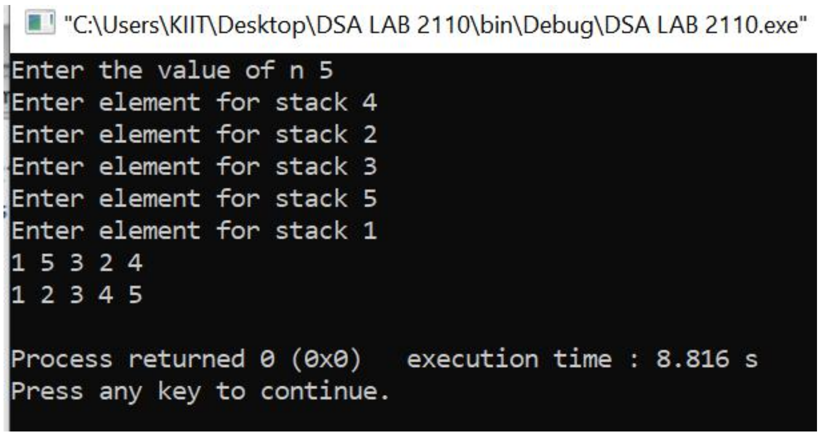
QUESTION NUMBER -3(STACKS)

➢ **Sorting a stack using a temporary stack**

```
#include <stdio.h>

#include <stdlib.h>


struct Stack

{

        int data;

        struct Stack *next;

}*input=NULL , *tmpStack=NULL;




void push(struct Stack **st,int x)

{

        struct Stack *t;

        t = (struct Stack *)malloc(sizeof(struct Stack));


        if (st == NULL)

        {

                printf("Stack is    full \n");

        }

        else
        {
                t->data =x;
                t->next = *st;
                *st = t;
        }
}
int pop(struct Stack    **st)
{
        struct Stack *t;
        t=(struct Stack *)malloc(sizeof(struct Stack));;
        int x;
        if (st == NULL)
        {
                printf("stack is empty");
        }
        else
        {
                t = *st;
                *st=(*st)->next;
                x = t->data;
                free(t);
        }
        return x;
}
```

```
void display(struct Stack *st)

{

    struct Stack *p;

    p = st;

    while (p)

    {

        printf("%d ", p->data);

        p = p->next;

    }

    printf("\n");

}


void sort(){

    while(input){

        int temp=pop(&input);

        while(tmpStack && tmpStack->data < temp ){

            push(&input,pop(&tmpStack));

        }

        push(&tmpStack,temp);

    }

}


void recursuion()

{
        while(input){

                int temp=pop(&input);

                push(&tmpStack,temp);

        }


}


void clone()

{

        while(tmpStack)

        {

                int temp=pop(&tmpStack);

                push(&input,temp);

        }

}




        int main(){

                int n;

                printf("Enter the value of n ");

                scanf("%d",&n);

        for (int i = 0; i < n; i++)

        {
```

```
        printf("Enter element for stack ");

        int a;

        scanf("%d",&a);

        push(&input,a);

    }

    display(input);

    sort();

    display(tmpStack);



    return 0;

}
```



```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"
Enter the value of n 5
Enter element for stack 4
Enter element for stack 2
Enter element for stack 3
Enter element for stack 5
Enter element for stack 1
1 5 3 2 4
1 2 3 4 5

Process returned 0 (0x0)   execution time : 8.816 s
Press any key to continue.
```

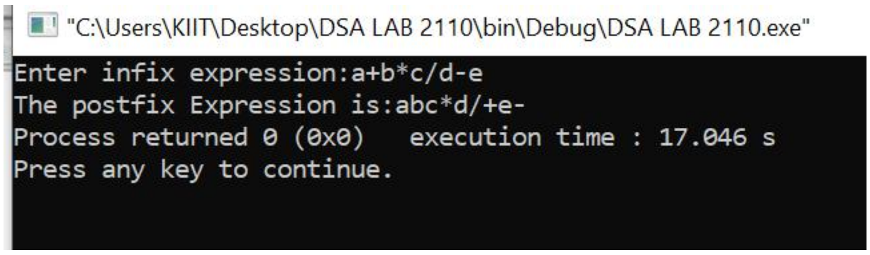
➢ Infix to postfix using a stack

```
#include<stdio.h>

#include<stdlib.h>

struct Node

{

        int data;

        struct Node *next;

};

typedef struct

{

        struct Node *top;

}Stack;

int IsEmpty(Stack s);

int push(Stack *s,int v);

int pop(Stack *s,struct Node **v);

void display(Stack s,int a,int n);

int IsOperand(int c);

int getVal(char a);

int IsLtoH(char a,char b);

int Infix_Postfix(char *inp , char *out);

struct Node *m;

int main()

{
        char inp[100];

        int l=0,i=0;

        printf("Enter infix expression:");

        gets(inp);

        while(inp[l] != '\0')

        {

            l++;

        }

        char out[l];

        printf("The postfix Expression is:");

        int c=Infix_Postfix(inp,out);

        while(out[i]!='\0')

        {

            printf("%c",out[i]);

            i++;

        }

        return 0;

}

int Infix_Postfix(char *inp , char *out)

{

        Stack s;

        s.top=NULL;

        int i=0,k=0,p,q;

        while(inp[i] != '\0')
```

```
{
        if(IsOperand(inp[i]))
        {
                out[k++]=inp[i];
        }
        else if(inp[i]==32)
        {
                i++;
                continue;
        }
        else if(inp[i]=='(')
        {
                q=push(&s,inp[i]);
        }
        else if(inp[i]==')')
        {
                while(1>0)
                {
                        p=pop(&s,&m);
                        if(p==1)
                        {
                                printf("Improper bracket pairs\n");
                                return 1;
                        }
                if(m->data=='(')
                    break;
                out[k++]=m->data;
            }
        }
        else
        {
            if(IsEmpty(s))
            {
                q=push(&s,inp[i]);
            }
            else
            {
                p=pop(&s,&m);
                if(p==1)
                {
                    printf("Improper bracket pairs\n");
                    return 1;
                }
                if((m->data=='(')||(IsLtoH(m->data,inp[i])))
                {
                    push(&s,m->data);
                    push(&s,inp[i]);
                }
                else
                {
                        out[k++]=m->data;
                        i--;
                }

            }
        }
        i++;
    }
    while(!(IsEmpty(s)))
    {
        p=pop(&s,&m);
        if(p==1)
        {
            printf("Improper bracket pairs\n");
            return 1;
        }
        out[k++]=m->data;
    }
    out[k]='\0';
}
int IsLtoH(char a,char b)
{
	if(getVal(a)<getVal(b))
		return 1;
	else
		return 0;
}
int getVal(char a)
{
	int t;
	switch(a)
	{
		case '+':
		case '-': t=1;
				break;
		case '*':
		case '/':t=2;
				break;
		case '^':t=3;
				break;
	}
	return t;
}
int IsOperand(int c)
{
	if( ((c>=65)&&(c<=90)) || ((c>=97)&&(c<=122)) )
	{
		return 1;
	}
	return 0;
}
int IsEmpty(Stack s)
{
	if(s.top==NULL)
		return 1;
	return 0;
}
int push(Stack *s,int v)
{
	struct Node *cur;
	cur= (struct Node *)malloc(sizeof(struct Node));
	if(cur==NULL)
	{
		printf("Overflow");
		return 1;
	}
	cur->data=v;
	cur->next = s->top;
	s->top=cur;
	return 0;
```

```
}
int pop(Stack *s,struct Node **v)
{
        if(IsEmpty(*s))
        {
                printf("Underflow");
                return 1;
        }
        *v = s->top;
        s->top=s->top->next;
        return 0;
}
void display(Stack s,int a,int n)
{
        Stack s1;
        s1.top=NULL;
        int k,l;
        while(!(IsEmpty(s)))
        {
                k=pop(&s,&m);
                if(a==1)
                        printf("%d\t",m->data);
                l=push(&s1,m->data);
        }
```

```
    while(!(IsEmpty(s1)))
    {
        k=pop(&s1,&m);
        if(a==0)
            printf("%d\t",m->data);
        l=push(&s,m->data);
    }
}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
Enter infix expression:a+b*c/d-e
The postfix Expression is:abc*d/+e-
Process returned 0 (0x0)   execution time : 17.046 s
Press any key to continue.
```

➢ Infix to prefix using a stack

```
#include<stdio.h>

#include<math.h>

#include<string.h>

#include <stdlib.h>

#define MAX 20

void push(int);

char pop();

void infix_to_prefix();

int precedence (char);

char stack[20],infix[20],prefix[20];

int top = -1;


int main()

{

printf("\nINPUT THE INFIX EXPRESSION : ");

scanf("%s",infix);

infix_to_prefix();

return 0;

}

void push(int pos)

{
```

```
if(top == MAX-1)

{

printf("\nSTACK OVERFLOW\n");

}

else {

top++;

stack[top] = infix[pos];

}}


char pop()

{

char ch;

if(top < 0)

{

printf("\nSTACK UNDERFLOW\n");

exit(0);

}

else

{

ch = stack[top];

stack[top] = '\0';

top--;

return(ch);

}
```

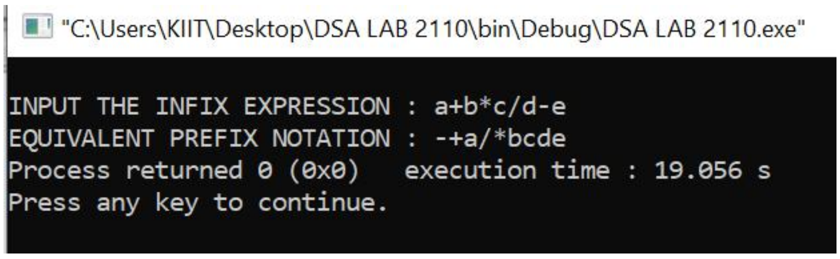
```
return 0;

}

void infix_to_prefix()

{

int i = 0,j = 0;

strrev(infix);

while(infix[i] != '\0')

{

if(infix[i] >= 'a' && infix[i] <= 'z')

{

prefix[j] = infix[i];

j++;

i++;

}

else if(infix[i] == ')' || infix[i] == '}' || infix[i] == ']')

{

push(i);

i++;

}

else if(infix[i] == '(' || infix[i] == '{' || infix[i] == '[')

{

if(infix[i] == '(')

{

while(stack[top] != ')')
```

```
{
prefix[j] = pop();
j++;
}
pop();
i++;
}
else if(infix[i] == '[')
{
while(stack[top] != ']')
{
prefix[j] = pop();
j++;
}
pop();
i++;
}
else if(infix[i] == '{')
{
while(stack[top] != '}')
{
prefix[j] = pop();
j++;
}
```

```
pop();

i++;

}}

else

{

if(top == -1)

{

push(i);

i++;

}

else if( precedence(infix[i]) < precedence(stack[top]))

{

prefix[j] = pop();

j++;

while(precedence(stack[top]) > precedence(infix[i])){

prefix[j] = pop();

j++;

if(top < 0) {

break;

}}

push(i);

i++;

}
```

```
else if(precedence(infix[i]) >= precedence(stack[top]))

{

push(i);

i++;

}}}

while(top != -1)

{

prefix[j] = pop();

j++;

}

strrev(prefix);

prefix[j] = '\0';

printf("EQUIVALENT PREFIX NOTATION : %s ",prefix);

}




int precedence(char alpha)

{

if(alpha == '+' || alpha =='-')

{

return(1);

}
```

```
if(alpha == '*' || alpha =='/')

{

return(2);

}

return 0;

}
```



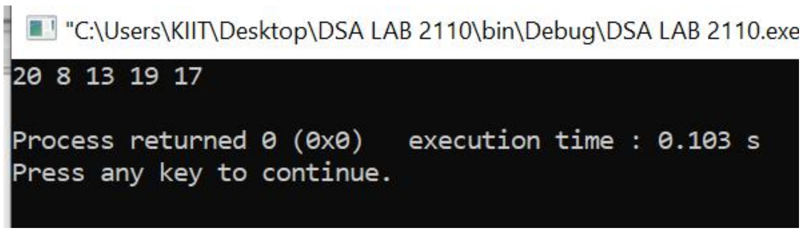
```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

INPUT THE INFIX EXPRESSION : a+b*c/d-e
EQUIVALENT PREFIX NOTATION : -+a/*bcde
Process returned 0 (0x0)   execution time : 19.056 s
Press any key to continue.
```

QUESTION NUMBER-4(QUEUES)

➢ Queue using a linked list

```
#include<stdio.h>

#include<stdlib.h>

#define m 5

struct node

{

	int data;

	struct node *nxt;

};

typedef struct

{

	struct node *r,*f;

}queue;


int insert(queue *q,int n)

{

	struct node *ptr=(struct node*)malloc(sizeof(struct node));

	if(ptr==NULL)

	{

	  printf("Full Queue.\n");

	  return 1;

	}
```

```
        ptr->data=n;

        ptr->nxt=NULL;

        if(q->r==NULL)

        {

          q->r=q->f=ptr;

        }

        else

        {

          q->r->nxt=ptr;

          q->r=ptr;

        }

        return 0;

}



int delete(queue *q,int *n)

{

        if(q->f==NULL)

        {

          printf("Empty Queue.\n");

          return 1;

        }

        if(q->f==q->r)

        {
          *n=q->f->data;
          free(q->f);
          q->r=q->f=NULL;
        }
        else
        {
          struct node *ptr=q->f;
          *n=q->f->data;
          q->f=q->f->nxt;
          free(ptr);
        }
        return 0;
}
int main()
{
        queue q;
        q.r=q.f=NULL;
        int n,x,y;
        for(int i=0; i<m; i++)
        {
          x=insert(&q,rand()%21);
        }
        for(int i=0; i<m; i++)
        {
          y=delete(&q,&n);

          printf("%d ",n);

        }

        printf("\n");

        return 0;

}
```

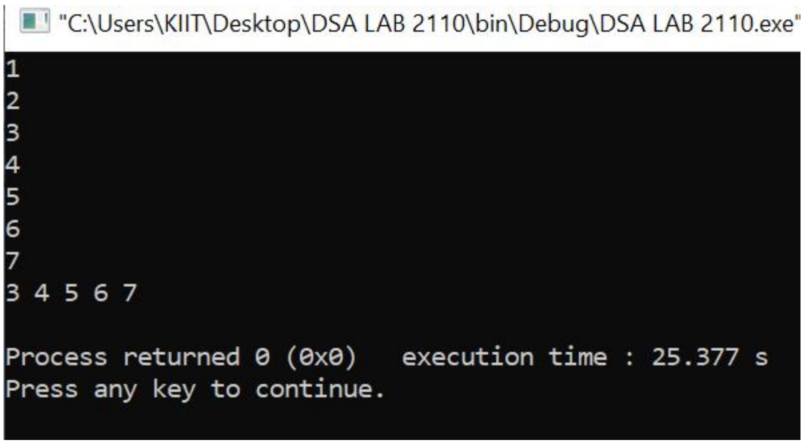


```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe
20 8 13 19 17

Process returned 0 (0x0)   execution time : 0.103 s
Press any key to continue.
```

➢ Circular queue using arrays

```
#include<stdio.h>

#include<stdlib.h>

#define m 5

typedef struct

{

        int data[m];

        int r,f;

}queue;

int insert(queue *q,int n)

{

        if((q->r==m-1 && q->f==0) || q->f==q->r+1)

        {

          printf("Insertion Not Possible.\n");

          return 1;

        }

        if(q->r==-1)

        {

          q->f=q->r=0;

          q->data[q->r]=n;

        }

        else

        {
          q->r=(q->r+1)%m;

          q->data[q->r]=n;

        }

        return 0;

}

int delete(queue *q, int *n)

{

        if(q->f==-1)

        {

          printf("Deletion Note Possible.\n");

          return 1;

        }

        if(q->r==q->f)

        {

          *n=q->data[q->f];

          q->f=q->r=-1;

        }

        else

        {

          *n=q->data[q->f];

          q->f=(q->f+1)%m;

        }

        return 0;

}
```

```
int main()

{

        queue q;

        q.f=q.r=-1;

        int n,x,y;

        for(int i=0; i<m-2;i++)

        {

          scanf("%d",&n);

          x=insert(&q,n);

        }

        y=delete(&q,&n);

        y=delete(&q,&n);

        for(int i=0; i<m-1; i++)

        {

          scanf("%d",&n);

          x=insert(&q,n);

        }

        for(int i=0; i<m; i++)

        {

          y=delete(&q,&n);

          printf("%d ",n);

        }

        printf("\n");

}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
1
2
3
4
5
6
7
3 4 5 6 7

Process returned 0 (0x0)   execution time : 25.377 s
Press any key to continue.
```

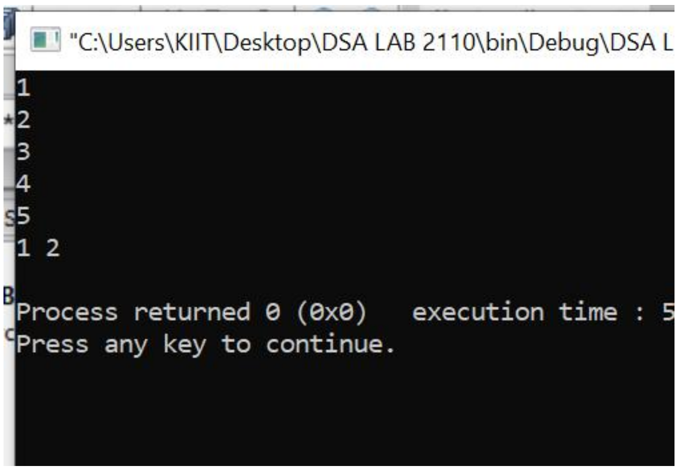
➢ Circular queue using Linked list

```
#include<stdio.h>

#include<stdlib.h>

#define m 5

struct node

{

	int data;

	struct node *nxt;

};

typedef struct

{

	struct node *r;

}queue;

int insert(queue *q, int n)

{

	struct node *ptr=(struct node*)malloc(sizeof(struct node));

	if(ptr==NULL)

	{

	  printf("Insetion Not Possible.\n");

	  return 1;

	}

	ptr->data=n;
```

```
        ptr->nxt=NULL;

        if(q->r==NULL)

        {

          q->r=ptr;

          q->r->nxt=q->r;

        }

        else

        {

          ptr->nxt=q->r->nxt;

          q->r->nxt=ptr;

          q->r=ptr;

        }

        return 0;

}

int delete(queue *q, int *n)

{

        if(q->r==NULL)

        {

          printf("Deletion Not Possible.\n");

          return 1;

        }

        if(q->r==q->r->nxt)

        {

          *n=q->r->data;
```

```
            free(q->r);
            q->r=NULL;
          }
          else
          {
            struct node *ptr=q->r->nxt;
            q->r->nxt=ptr->nxt;
            *n=ptr->data;
            free(ptr);
          }
          return 0;
}
int main()
{
          queue q;
          q.r=NULL;
          int n,x,y;
          for(int i=0; i<m;i++)
          {
            scanf("%d",&n);
            x=insert(&q,n);
          }
          y=delete(&q,&n);
          printf("%d ",n);
```

```
        y=delete(&q,&n);

        printf("%d ",n);

        printf("\n");

}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA L

```
1
2
3
4
5
1 2

Process returned 0 (0x0)   execution time : 5
Press any key to continue.
```

➢ Implement stack using a queue

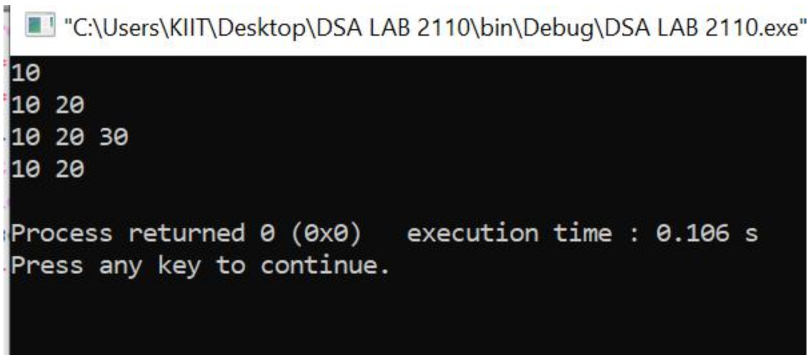
```
#include<stdio.h>

#include<stdlib.h>

#define MAX 20


typedef struct{

        int data[MAX];

        int front;

        int rear;

}Queue;


Queue q1;


int insert(Queue *q, int v){

        if(q->rear == MAX-1){        //full Q

                printf("Queue is full\n");

                return 1;

        }

        if(q->rear == -1){      //Empty Q

                q->front = q->rear = 0;

                q->data[q->rear] = v;

        }else{      //Partially full Q
```

```
		q->rear++;

		q->data[q->rear] = v;

	}

	return 0;

}


int delete(Queue *q, int *m){

	if(q->front == -1){	//Empty Q

		printf("Q is empty\n");

		return 1;

	}

	if(q->front == q->rear){

		*m = q->data[q->front];

		q->front = q->rear = -1;

	}else{

		*m = q->data[q->front];

		q->front++;

	}

	return 0;

}


int delete2(Queue *q, int *m){

	if(q->front == -1){

		printf("Q is empty\n");
```

```
        return 1;
    }
    if(q->front == q->rear){
        *m = q->data[q->front];
        q->front = q->rear = -1;
    }else{
        *m = q->data[q->front];
        for(int i=1; i<q->rear; i++){
            q->data[i-1] = q->data[i];
        }
        q->rear--;
    }
    return 0;
}

void display(Queue q){
    int i;
    if (q.front == - 1)
        printf("Queue is empty \n");
    else{
        for (i = q.front; i <= q.rear; i++)
            printf("%d ", q.data[i]);
        printf("\n");
    }
```

```
}


int isEmpty(Queue q){
	return (q.front == -1) ? 1 : 0;
}


int push(int v){
	return insert(&q1, v);
}


int pop(int *m){
	int p = isEmpty(q1);
	if(p) return p;
	int i = q1.front;
	int j = q1.rear;
	while(i!=j){
		int n;
		delete(&q1, &n);
		insert(&q1, n);
		i++;
	}
	int n;
	delete(&q1, &n);
	*m = n;
```

```
        return 0;

}


int main(){

        q1.front = q1.rear = -1;

        int t = push(10);

        display(q1);

        t = push(20);

        display(q1);

        t = push(30);

        display(q1);

        int m;

        int r = pop(&m);

        display(q1);

        return 0;

}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
10
10 20
10 20 30
10 20

Process returned 0 (0x0)   execution time : 0.106 s
Press any key to continue.
```

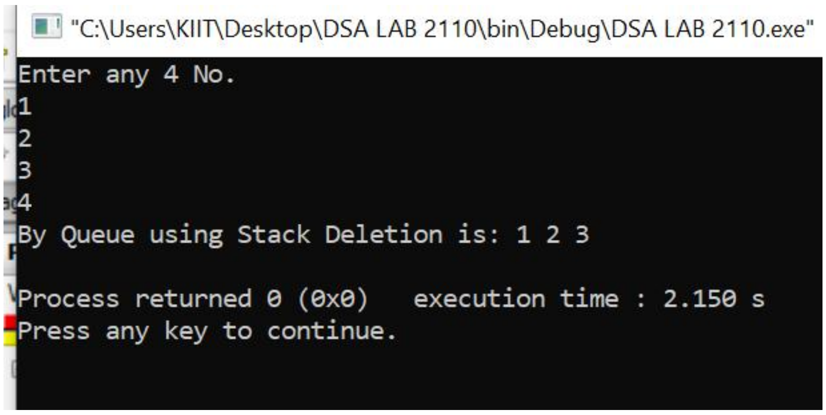
➢ Implement queue using a stack

```
#include <stdio.h>

#include <stdlib.h>


struct node {

        int data;

        struct node* next;

};


void push(struct node** top_ref, int new_data);


int pop(struct node** top_ref);


struct queue {

        struct node* stack1;

        struct node* stack2;

};


void enQueue(struct queue* q, int x)

{

        push(&q->stack1, x);

}
```

```
int deQueue(struct queue* q)
{
    int x;



    if (q->stack1 == NULL && q->stack2 == NULL)
    {
        printf("Q is empty");
        getchar();
        exit(0);
    }

    if (q->stack2 == NULL)
    {
        while (q->stack1 != NULL)
        {
            x = pop(&q->stack1);
            push(&q->stack2, x);
        }
    }

    x = pop(&q->stack2);
    return x;
}
```

```
void push(struct node** top_ref, int new_data)

{

    struct node* new_node = (struct node*)malloc(sizeof(struct node));

    if (new_node == NULL) {

        printf("Stack overflow \n");

        getchar();

        exit(0);

    }


    new_node->data = new_data;

    new_node->next = (*top_ref);

    (*top_ref) = new_node;

}


int pop(struct node** top_ref)

{

    int res;

    struct node* top;


    if (*top_ref == NULL) {

        printf("Stack underflow \n");

        getchar();

        exit(0);
        }
        else {
                top = *top_ref;
                res = top->data;
                *top_ref = top->next;
                free(top);
                return res;
        }
}

int main()
{
        struct queue* q = (struct queue*)malloc(sizeof(struct queue));
        q->stack1 = NULL;
        q->stack2 = NULL;
        printf("Enter any 4 No.\n");
        int n;
        scanf("%d",&n);
        enQueue(q, n);
        scanf("%d",&n);
        enQueue(q, n);
        scanf("%d",&n);
        enQueue(q, n);
        scanf("%d",&n);
```

```
    enQueue(q, n);

    printf("By Queue using Stack Deletion is: ");

    printf("%d ", deQueue(q));

    printf("%d ", deQueue(q));

    printf("%d ", deQueue(q));

    printf("\n");

    return 0;

}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
Enter any 4 No.
1
2
3
4
By Queue using Stack Deletion is: 1 2 3

Process returned 0 (0x0)   execution time : 2.150 s
Press any key to continue.
```

QUESTION NUMBER-5 (LINKED LIST)

➢ **Remove duplicates from a linked list**

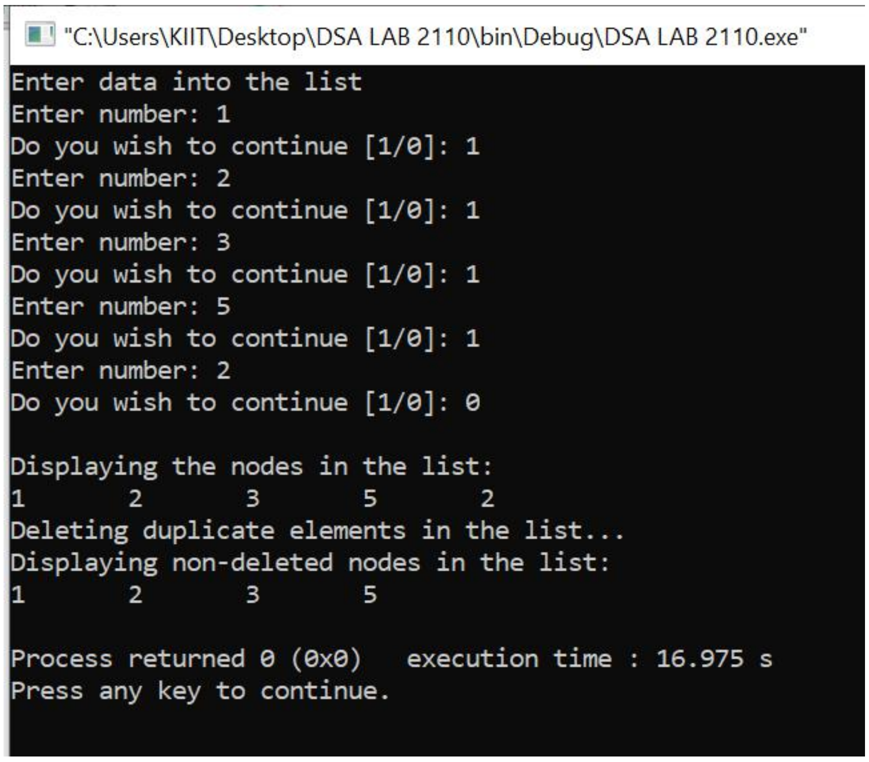
```
#include <stdio.h>

#include <stdlib.h>


struct node

{

        int num;

        struct node *next;

};


void create(struct node **);

void dup_delete(struct node **);

void release(struct node **);

void display(struct node *);


int main()

{

        struct node *p = NULL;

        struct node_occur *head = NULL;

        int n;
```

```
    printf("Enter data into the list\n");

    create(&p);

    printf("Displaying the nodes in the list:\n");

    display(p);

    printf("Deleting duplicate elements in the list...\n");

    dup_delete(&p);

    printf("Displaying non-deleted nodes in the list:\n");

    display(p);

    release(&p);


    return 0;

}


void dup_delete(struct node **head)

{

    struct node *p, *q, *prev, *temp;


    p = q = prev = *head;

    q = q->next;

    while (p != NULL)

    {

        while (q != NULL && q->num != p->num)

        {
```

```
                        prev = q;
                        q = q->next;
                }
                if (q == NULL)
                {
                        p = p->next;
                        if (p != NULL)
                        {
                                q = p->next;
                        }
                }
                else if (q->num == p->num)
                {
                        prev->next = q->next;
                        temp = q;
                        q = q->next;
                        free(temp);
                }
        }
}

void create(struct node **head)
{
        int c, ch;
```

```
    struct node *temp, *rear;


    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    } while (ch != 0);
    printf("\n");
}
```

```
void display(struct node *p)
{
    while (p != NULL)
    {
        printf("%d\t", p->num);
        p = p->next;
    }
    printf("\n");
}

void release(struct node **head)
{
    struct node *temp = *head;
    *head = (*head)->next;
    while ((*head) != NULL)
    {
        free(temp);
        temp = *head;
        (*head) = (*head)->next;
    }
}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
Enter data into the list
Enter number: 1
Do you wish to continue [1/0]: 1
Enter number: 2
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 5
Do you wish to continue [1/0]: 1
Enter number: 2
Do you wish to continue [1/0]: 0

Displaying the nodes in the list:
1       2       3       5       2
Deleting duplicate elements in the list...
Displaying non-deleted nodes in the list:
1       2       3       5

Process returned 0 (0x0)   execution time : 16.975 s
Press any key to continue.
```

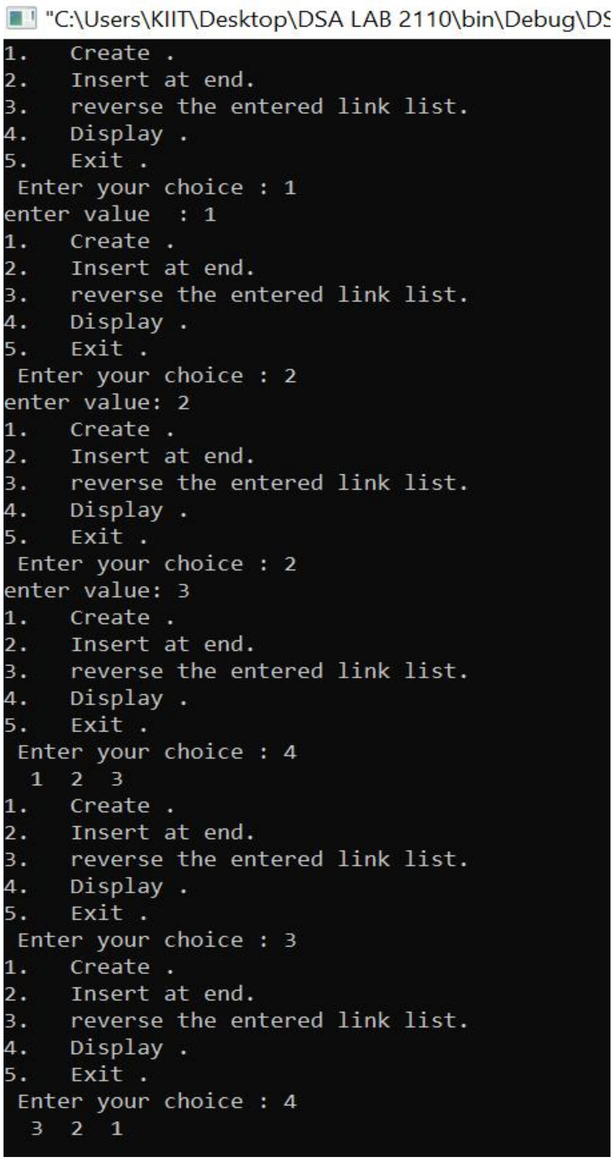
➢ Reverse a linked list

```
#include<stdio.h>

#include<stdlib.h>


struct node{

        int data;

        struct node *next;

};


struct node *start = NULL;


struct node *create(struct node *start){

        int n;

        struct node * new_node = (struct node *)malloc(sizeof(struct node));

        printf("enter value    : ");

        scanf("%d",&n);

        new_node->data =n;

        new_node->next =start;

        start=new_node;

        return start;

}
```

```
struct node *ins_end(struct node* start){

        int n;

        struct node *new_node = (struct node *)malloc(sizeof(struct node));

        struct node *ptr;

        printf("enter value: ");

        scanf("%d",&n);

        new_node->data =n;

        new_node->next =NULL;

        ptr = start;

        while(ptr->next !=NULL){

          ptr =ptr->next;

        }

        ptr->next= new_node;

        return start;

}


struct node *rev(struct node*start){



struct node *ptr,*preptr=NULL,*temp;

        ptr=start;

        while(ptr!=NULL){

          temp=ptr->next;

          ptr->next=preptr;

          preptr =ptr;
```

```
              ptr=temp;
        }
            start= preptr;
            return start;
}



void display(struct node *start){
            struct node *temp;
            if(start == NULL)
              printf("the linked list doesn't exists. ");
            else{
              temp = start;
              while(temp != NULL){
                     printf("    %d",temp->data);
                     temp = temp->next;
              }
              printf("\n");
            }
}
```

```
int main(){
      int ch;
      while(1){
        printf("1.     Create .\n");
        printf("2.     Insert at end.\n");
        printf("3.     reverse the entered link list.\n");
        printf("4.     Display .\n");
        printf("5.     Exit .\n");
        printf(" Enter your choice : ");
        scanf("%d",&ch);
        switch(ch){
            case 1:
                start = create(start);
                break;
            case 2:
                start = ins_end(start);
                break;
            case 3:
                start = rev(start);
                break;
            case 4:
                display(start);
                break;
            case 5:
                    exit(0);

                    break;

                default:

                    printf(" wrong choice . \n");

         }

        }

        return 0;

}
```



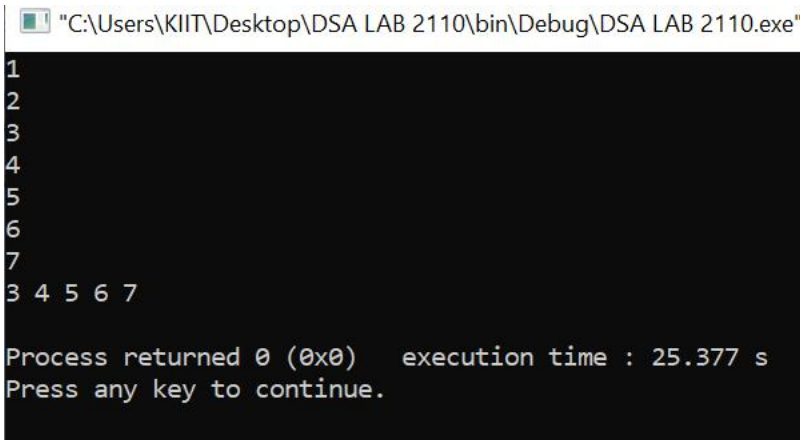
```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DS
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 1
enter value  : 1
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 2
enter value: 2
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 2
enter value: 3
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 4
  1  2  3
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 3
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 4
  3  2  1
```

➢ Circular queue using arrays

```
#include<stdio.h>

#include<stdlib.h>

#define m 5

typedef struct

{

        int data[m];

        int r,f;

}queue;

int insert(queue *q,int n)

{

        if((q->r==m-1 && q->f==0) || q->f==q->r+1)

        {

          printf("Insertion Not Possible.\n");

          return 1;

        }

        if(q->r==-1)

        {

          q->f=q->r=0;

          q->data[q->r]=n;

        }

        else

        {
```

```
			q->r=(q->r+1)%m;

			q->data[q->r]=n;

		}

		return 0;

}

int delete(queue *q, int *n)

{

		if(q->f==-1)

		{

			printf("Deletion Note Possible.\n");

			return 1;

		}

		if(q->r==q->f)

		{

			*n=q->data[q->f];

			q->f=q->r=-1;

		}

		else

		{

			*n=q->data[q->f];

			q->f=(q->f+1)%m;

		}

		return 0;

}
```

```
int main()

{

        queue q;

        q.f=q.r=-1;

        int n,x,y;

        for(int i=0; i<m-2;i++)

        {

          scanf("%d",&n);

          x=insert(&q,n);

        }

        y=delete(&q,&n);

        y=delete(&q,&n);

        for(int i=0; i<m-1; i++)

        {

          scanf("%d",&n);

          x=insert(&q,n);

        }

        for(int i=0; i<m; i++)

        {

          y=delete(&q,&n);

          printf("%d ",n);

        }

        printf("\n");

}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
1
2
3
4
5
6
7
3 4 5 6 7

Process returned 0 (0x0)   execution time : 25.377 s
Press any key to continue.
```

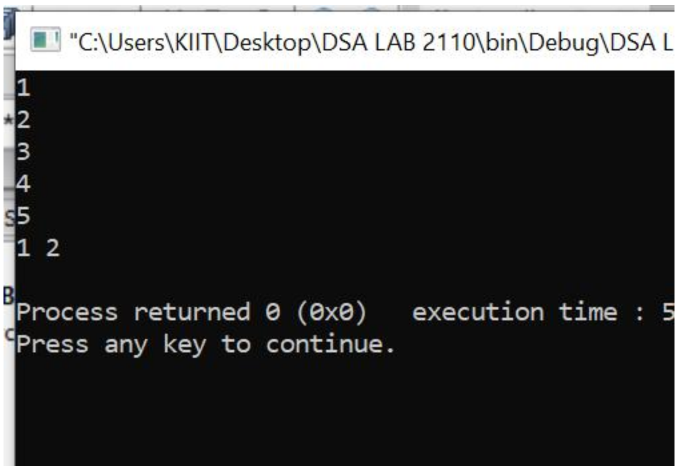
➢ Circular queue using Linked list

```
#include<stdio.h>

#include<stdlib.h>

#define m 5

struct node

{

	int data;

	struct node *nxt;

};

typedef struct

{

	struct node *r;

}queue;

int insert(queue *q, int n)

{

	struct node *ptr=(struct node*)malloc(sizeof(struct node));

	if(ptr==NULL)

	{

	  printf("Insetion Not Possible.\n");

	  return 1;

	}

	ptr->data=n;
```

```
        ptr->nxt=NULL;
        if(q->r==NULL)
        {
          q->r=ptr;
          q->r->nxt=q->r;
        }
        else
        {
          ptr->nxt=q->r->nxt;
          q->r->nxt=ptr;
          q->r=ptr;
        }
        return 0;
}
int delete(queue *q, int *n)
{
        if(q->r==NULL)
        {
          printf("Deletion Not Possible.\n");
          return 1;
        }
        if(q->r==q->r->nxt)
        {
          *n=q->r->data;
```

```
              free(q->r);
              q->r=NULL;
            }
            else
            {
              struct node *ptr=q->r->nxt;
              q->r->nxt=ptr->nxt;
              *n=ptr->data;
              free(ptr);
            }
            return 0;
}
int main()
{
            queue q;
            q.r=NULL;
            int n,x,y;
            for(int i=0; i<m;i++)
            {
              scanf("%d",&n);
              x=insert(&q,n);
            }
            y=delete(&q,&n);
            printf("%d ",n);
```

```
        y=delete(&q,&n);

        printf("%d ",n);

        printf("\n");

}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA L

```
1
2
3
4
5
1 2

Process returned 0 (0x0)   execution time : 5
Press any key to continue.
```

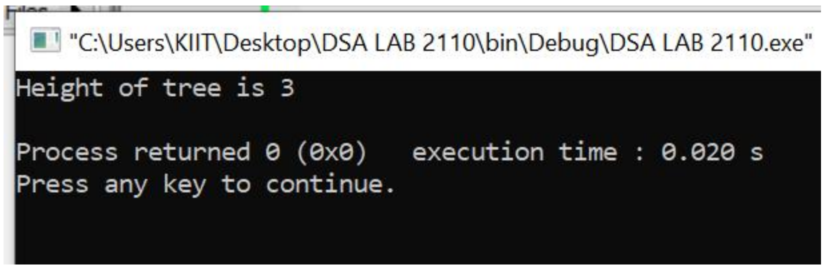
QUESTION NUMBER - 6(TREES)

➢ Height of a binary tree

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

        int data;

        struct node* left;

        struct node* right;

};


int maxDepth(struct node* node)

{

      if (node==NULL)

              return 0;

      else

      {

              int lDepth = maxDepth(node->left);

              int rDepth = maxDepth(node->right);
```

```
           if (lDepth > rDepth)
                 return(lDepth+1);
           else return(rDepth+1);
     }
}

struct node* newNode(int data)
{
     struct node* node = (struct node*)
     malloc(sizeof(struct node));
     node->data = data;
     node->left = NULL;
     node->right = NULL;

     return(node);
}

int main()
{
     struct node *root = newNode(1);

     root->left = newNode(2);
     root->right = newNode(3);
     root->left->left = newNode(4);
```

```
    root->left->right = newNode(5);


    printf("Height of tree is %d\n", maxDepth(root));


    return 0;
}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

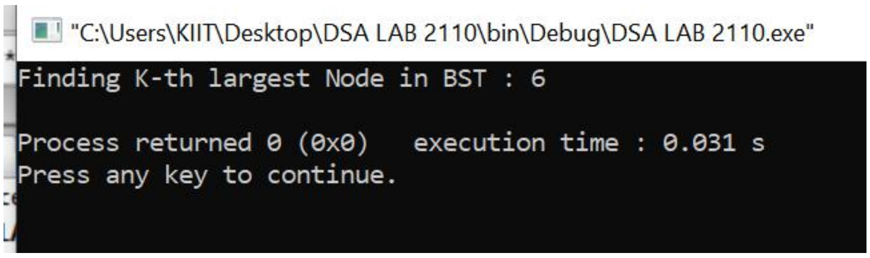
```
Height of tree is 3

Process returned 0 (0x0)   execution time : 0.020 s
Press any key to continue.
```

➢ Find kth maximum value in a binary search tree

```
#include <stdio.h>


struct Node {
    int data;
    struct Node *left, *right;
};


struct Node* newNode(int data)
{
    struct Node* temp = malloc(sizeof(struct Node));
    temp->data = data;
    temp->right = temp->left = NULL;
    return temp;
}


struct Node* KthLargestUsingMorrisTraversal(struct Node* root, int k)
{
    struct Node* curr = root;
    struct Node* Klargest = NULL;


    int count = 0;
    while (curr != NULL) {
        if (curr->right == NULL) {

            if (++count == k)
                Klargest = curr;

            curr = curr->left;
        }

        else {

            struct Node* succ = curr->right;

            while (succ->left != NULL && succ->left != curr)
                succ = succ->left;

            if (succ->left == NULL) {

                succ->left = curr;

                curr = curr->right;
            }

            else {
```

```
                        succ->left = NULL;

                        if (++count == k)
                              Klargest = curr;

                        curr = curr->left;
                  }
            }
      }

      return Klargest;
}

int main()
{
      /* Constructed binary tree is
               4
            /     \
           2         7
        /   \     /   \
      1       3    6       10 */

      struct Node* root = newNode(4);
```

```
    root->left = newNode(2);

    root->right = newNode(7);

    root->left->left = newNode(1);

    root->left->right = newNode(3);

    root->right->left = newNode(6);

    root->right->right = newNode(10);



    printf("Finding K-th largest Node in BST : %d\n", KthLargestUsingMorrisTraversal(root, 3)->data);



    return 0;

}
```

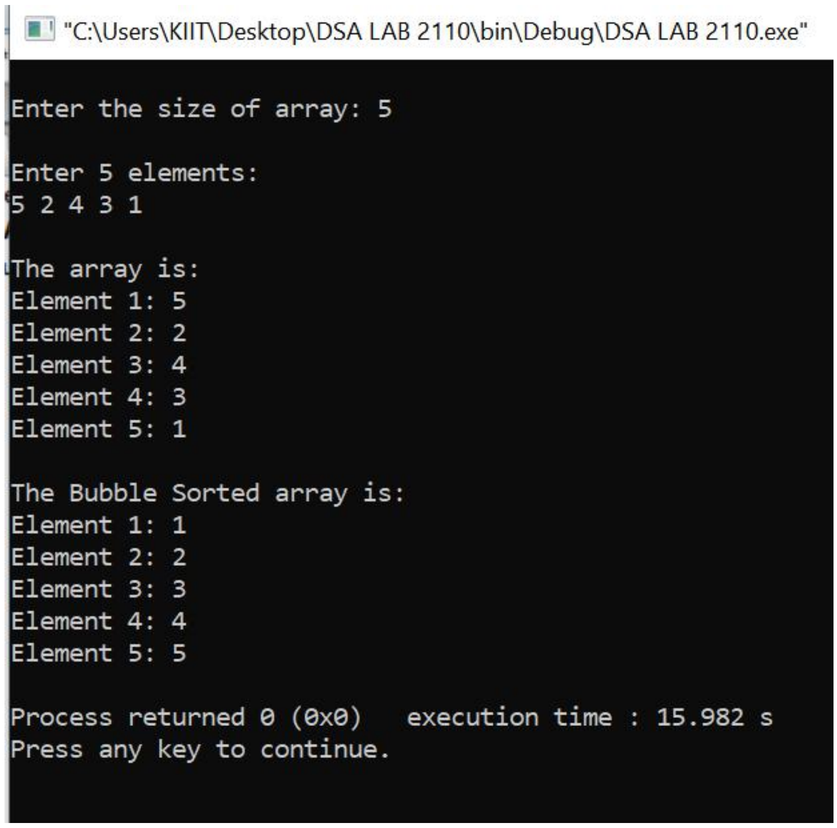


```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"
Finding K-th largest Node in BST : 6

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

QUESTION NUMBER -7 (SEARCHING AND SORTING)

- **Bubble Sort**

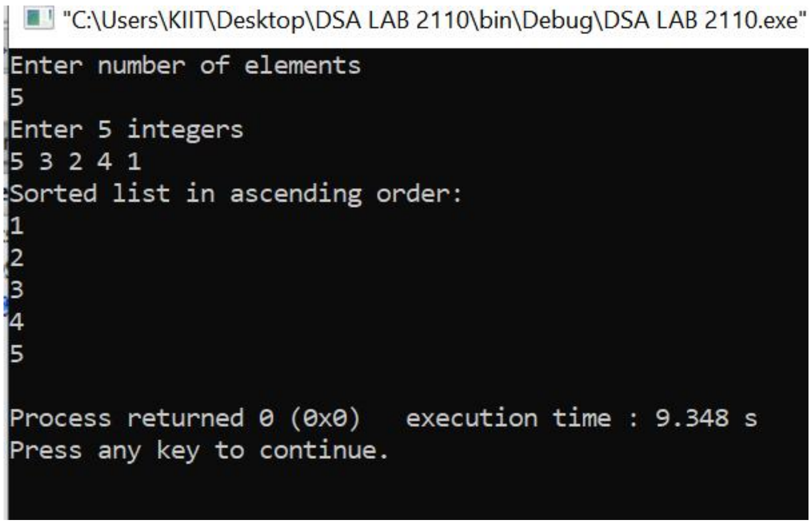
```
#include <stdio.h>

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

        printf("\nThe array is:\n");
        for (int i = 0; i < size; i++)
          printf("Element %d: %d\n", i + 1, array[i]);

        for (int i = 0; i < size; i++)
          for (int j = 0; j < size - 1; j++)
                if (array[j] > array[j + 1])
                {
                        int temp = array[j];
                        array[j] = array[j + 1];
                        array[j + 1] = temp;
                }

        printf("\nThe Bubble Sorted array is:\n");
        for (int i = 0; i < size; i++)
          printf("Element %d: %d\n", i + 1, array[i]);

        return 0;
}
```



```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

Enter the size of array: 5

Enter 5 elements:
5 2 4 3 1

The array is:
Element 1: 5
Element 2: 2
Element 3: 4
Element 4: 3
Element 5: 1

The Bubble Sorted array is:
Element 1: 1
Element 2: 2
Element 3: 3
Element 4: 4
Element 5: 5

Process returned 0 (0x0)   execution time : 15.982 s
Press any key to continue.
```

- **Selection Sort**

```
#include <stdio.h>

int main()

{

int array[100], n, c, d, position, t;

printf("Enter number of elements\n");

scanf("%d", &n);

printf("Enter %d integers\n", n);

for (c = 0; c < n; c++)

scanf("%d", &array[c]);

for (c = 0; c < (n - 1); c++)

{position = c;

for (d = c + 1; d < n; d++)

{

if (array[position] > array[d])

position = d;

}

if (position != c)

{

t = array[c];

array[c] = array[position];

array[position] = t;
}}

printf("Sorted list in ascending order:\n");

for (c = 0; c < n; c++)

printf("%d\n", array[c]);

return 0;

}
```



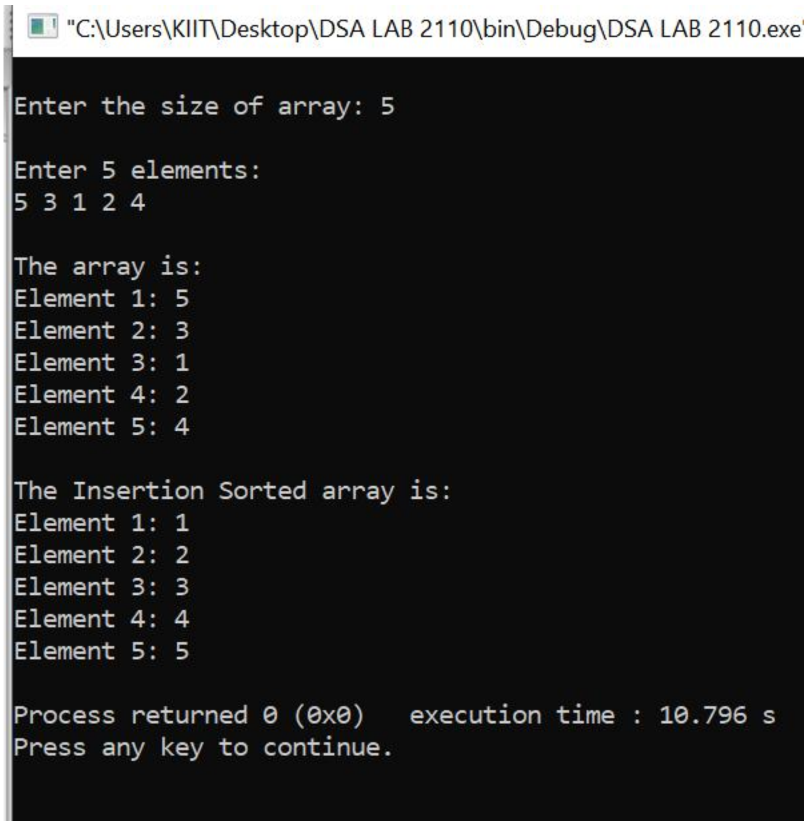
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
Enter number of elements
5
Enter 5 integers
5 3 2 4 1
Sorted list in ascending order:
1
2
3
4
5

Process returned 0 (0x0)   execution time : 9.348 s
Press any key to continue.
```

➢ Insertion Sort

```
#include <stdio.h>


int main()

{

	int size;

	printf("\nEnter the size of array: ");

	scanf("%d", &size);

	int array[size];

	printf("\nEnter %d elements:\n", size);

	for (int i = 0; i < size; i++)

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



	printf("\nThe array is:\n");

	for (int i = 0; i < size; i++)

	  printf("Element %d: %d\n", i + 1, array[i]);



	for (int i = 1; i <= size - 1; i++)

	{

	  int j = i;

	  while (j > 0 && array[j - 1] > array[j])

	  {

		int temp = array[j];
```

```
                array[j] = array[j - 1];

                array[j - 1] = temp;

                j--;

        }

      }



      printf("\nThe Insertion Sorted array is:\n");

      for (int i = 0; i < size; i++)

        printf("Element %d: %d\n", i + 1, array[i]);



      return 0;

}
```



```
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe

Enter the size of array: 5

Enter 5 elements:
5 3 1 2 4

The array is:
Element 1: 5
Element 2: 3
Element 3: 1
Element 4: 2
Element 5: 4

The Insertion Sorted array is:
Element 1: 1
Element 2: 2
Element 3: 3
Element 4: 4
Element 5: 5

Process returned 0 (0x0)   execution time : 10.796 s
Press any key to continue.
```

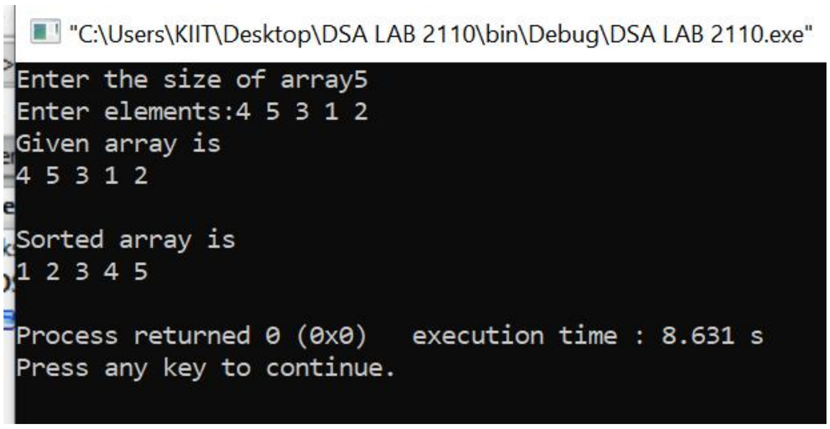
➢ Merge Sort

```
#include <stdio.h>

#include <stdlib.h>

void merge(int arr[], int l, int m, int r)

{

    int i, j, k;

    int n1 = m - l + 1;

    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)

        L[i] = arr[l + i];

    for (j = 0; j < n2; j++)

        R[j] = arr[m + 1 + j];

    i = 0;

    j = 0;

    k = l;

    while (i < n1 && j < n2) {

        if (L[i] <= R[j]) {

            arr[k] = L[i];

            i++;

        }

        else {

            arr[k] = R[j];
```

```
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
}

void mergeSort(int arr[], int l, int r)
{
        if (l < r) {
            int m = l + (r - l) / 2;
            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);

            merge(arr, l, m, r);
```

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

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

	mergeSort(arr, 0, arr_size - 1);
```

```
        printf("\nSorted array is \n");

        printArray(arr, arr_size);

        return 0;

}
```



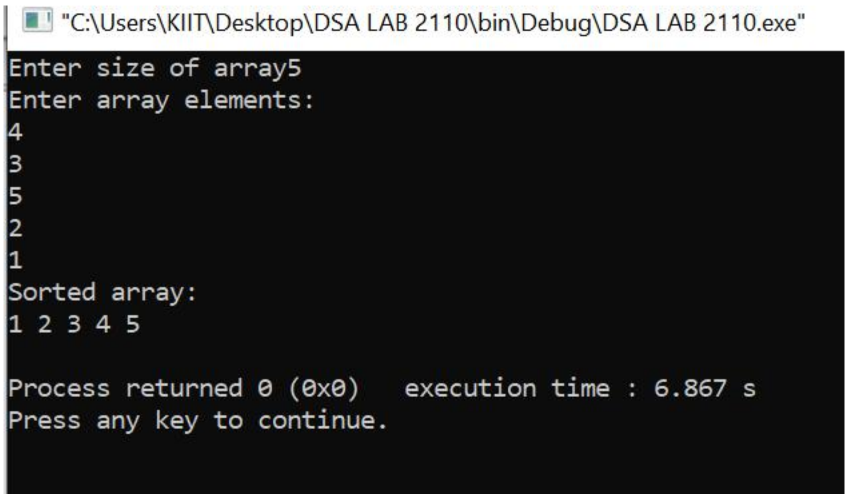
"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
Enter the size of array5
Enter elements:4 5 3 1 2
Given array is
4 5 3 1 2

Sorted array is
1 2 3 4 5

Process returned 0 (0x0)   execution time : 8.631 s
Press any key to continue.
```

➢ Quick Sort

```
#include<stdio.h>

void swap(int* a, int* b)

{

        int t = *a;

        *a = *b;

        *b = t;

}

int partition (int arr[], int low, int high)

{

        int pivot = arr[high];

        int i = (low - 1);

        for (int j = low; j <= high- 1; j++)

        {

                if (arr[j] < pivot)

                {

                        i++;

                        swap(&arr[i], &arr[j]);

                }

        }

        swap(&arr[i + 1], &arr[high]);

        return (i + 1);

}
```

```
void quickSort(int arr[], int low, int high)
{
        if (low < high)
        {
                int pi = partition(arr, low, high);


                quickSort(arr, low, pi - 1);
                quickSort(arr, pi + 1, high);
        }
}

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

int main()
{
        int n,i;
        printf("Enter size of array");
        scanf("%d",&n);

        int arr[n];

        printf("Enter array elements:\n");

        for(i=0;i<n;i++)

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

        quickSort(arr, 0, n-1);

        printf("Sorted array: \n");

        printArray(arr, n);

        return 0;

}
```



"C:\Users\KIIT\Desktop\DSA LAB 2110\bin\Debug\DSA LAB 2110.exe"

```
Enter size of array5
Enter array elements:
4
3
5
2
1
Sorted array:
1 2 3 4 5

Process returned 0 (0x0)   execution time : 6.867 s
Press any key to continue.
```

CHAUDHARY HAMDAN

1905387
DSA LAB
Date : 09-11-2020

Faculty in charge : Meghna Ma'am