# TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY

Name ..... Branch ..... Sem .....

Roll No. ..... Date .....

## INDEX

| S. No. | Experiment | Date | Remarks |
|---|---|---|---|
| 1. | WAP to insert an element in 1-D array. | 22-9-21 | |
| 2. | WAP to delete an element in 1D array. | 24-9-21 | |
| 3. | WAP to perform linear search operation on 1-D array | 27-9-21 | |
| 4. | WAP to perform binary search operation on 1-D array. | 27-9-21 | |
| 5. | WAP to implement bubble sort. | 28-9-21 | |
| 6. | WAP to implement selection sort | 30-9-21 | |
| 7. | WAP to implement insertion sort | 30-9-21 | |
| 8. | WAP to perform stack operation push, pop, display, peek. (Array) | 4-10-21 | |
| 9. | WAP to convert infix exp. to postfix exp. | 5-10-21 | |
| 10. | WAP to reverse a string using stack | 7-10-21 | |
| 11. | WAP to check balanced bracket with stack | 9-10-21 | |
| 12. | WAP to implement queue, dequeue & display operation on Queue. (Array) | 9-10-21 | |
| 13. | WAP to implement front insertion restricted Double Ended Queue. | 11-10-21 | |
| 14. | WAP to implement back deletion restricted Double Ended Queue. | 12-10-21 | |
| 15. | WAP to perform insertion in linked list. | 13-10-21 | |
| 16. | WAP to perform deletion in linked list. | 14-10-21 | |
| 17. | WAP to implement stack using linked list | 15-10-21 | |
| 18. | WAP to implement queue using array. | 18-10-21 | |
| 19. | WAP to implement queue with liked list | 16-10-21 | |
| 20. | WAP to implement circular queue with linked list | 18-10-21 | |

OUTPUT:

Where do you want to insert the element → 3
Enter the element → 7

1
2
7
3
4
5

# Experiment -1

Aim Write a programm to insert an element in 1-D array.

```
#include <stdio.h>
int main (){
        int arr [6]={1, 2, 3, 4,5};
        int pos, num, i;
        printf(" Where do you want to insert the elemen
        scanf("%d", &pos);
        printf(" Enter the element: ");
        scanf ("%d", &num);
        for(i=5; i>=(pos-1); i--){
                arr[i]= arr[i-1];
        }
        arr[pos-1]=num;
        for (i=0; i<6; i++)
                printf("%d\n", arr[i]);

        return 0;
}
```

OUTPUT:

The elements of the array are:
1
2
3
4
5

Enter the location of the element to be deleted →
1
2
4
5

# Experiment -2

Aim Write a program to delete an element from 1-D array

```
#include <stdio.h>
int main (){
        int loc, n=5, arr [5]={1, 2, 3, 4, 5}, i;
        printf ("The elements of the array are :" };
        for (i=0; i<n; i++) {
                printf ("%d\n", arr [i]);
        }
        printf ("Enter the location of the element to be delete
        scanf ("%d", &loc);
        for (i=0; i<n; i++)
                arr [i-1]= arr [i];
        n--;

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

        return 0;
}
```

OUTPUT:

Enter the number you want to find: → $\boxed{4}$

Element found at 10.

# Experiment - 3

**Aim** WAP to perform linear search operation on 1-D array.

```
#include <stdio.h>
void main(){
    int arr[12]={1,5,3,7,9,2,5,7,6,4,3,1}, i, ele, flag=1;
    printf("Enter the element to find:");
    scanf("%d", &ele);
    for(i=0; i<12; i++){
        if (ele == arr[i]){
            printf("Element found at %d", i+1);
            flag=1;
            break;
        }
    }
    if (flag>0)
        printf("Element not found");
}
```

OUTPUT:

Enter the element to find : → $\boxed{6}$

6 found at 6 position.

# Experiment - 4

Aim WAP to perform binary search operation on 1-D array.

```
#include <stdio.h>
void main (){
    int arr[10]={1,2,3,4,5,6,7,8,9,10}, ele=0, i,
    low=0, high=9, mid;
    printf("Enter the element to find :");
    scanf("%d", &ele);
    while (low <= high) {
        mid = (low + high)/2;
        if (arr[mid] == ele) {
            printf("%d found at %d position", ele,
                                                mid+1);

            break;
        }
        else if (arr[mid] > ele)
            high = mid -1;
        else
            low = mid +1;
    }
    if (low > high)
        printf("\n Not found");
}
```

OUTPUT:

Sorted elements are:

3 5 6 7 12

# TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY

Name _____ Branch _____ Sem _____

Roll No. _____ Date _____

## Experiment - 5

Aim WAP to implement bubble sort algorithm.

```
#include <stdio.h>
void main (){
    int a[10]={12,5,3,7,6}, n=5, i, j;
    for (j=0; j<n-1; j++){
        for (i=0; i<n-1-j; i++){
            if (a[i]> a[i+1]{
                int temp;
                temp =a[i];
                a[i]= a[i+1];
                a[i+1]= temp;
            }
        }
    } printf(" Sorted elements are: ");
    for(i=0; i<n; i++)
        printf("%d  ", a[i]);
}
```

OUTPUT

Sorted elements are:

0 2 6 11 12 19 34 45 55 99

OUTPUT:

Sorted elements are:
0 2 6 11 12 18 34 45 55 99

# Experiment - 6

Aim WAP to implement selection sort algorithm.

```
#include <stdio.h>
int main(){
    int arr[10]={6,12,0,18,11,99,55,45,34,2};
    int n=10;
    int i,j,position,swap;
    for(i=0;i<(n-1);i++){
        position=i;
        for(j=i+1;j<n;j++){
            if(arr[position]>arr[j])
                position=j;
        }
        if(position != i){
            swap=arr[i];
            arr[i]=arr[position];
            arr[position]=swap;
        }
    } printf("Sorted elements are:");
    for(i=0;i<n;i++)
        printf("%d ",arr[i]);
    return 0;
}
```

OUTPUT:

Sorted elements:

0 1 2 3 4 5 6 7 8 9

# Experiment - 7

**Aim** WAP to perform insertion sort algorithm.

```
#include <stdio.h>
int main (){
    int i, j, count =10, temp, number[10] ={4, 3, 5, 6,
                                            8, 1, 7, 9, 0, 2};

    ~~for (i=0;~~
    for (i=1; i< count; i++){
        temp = number [i];
        j= i-1;
        while ((temp< number[j]) && (j>=0)){
            number[j+1] = number [j];
            j=j-1;
        }
        number [j+1] = temp;
    }
    printf (" Sorted elements : ");
    for (i=0; i<count; i++)
        printf ("%d", number [i]);

    return 0;
}
```

OUTPUT:

Choices:
1) Push
2) Pop
3) Display
4) Peek
5) Exit

Enter the choice: → 1
Enter element: → 10

Enter the choice: → 2

Enter the choice: → 3
Underflow

Enter the choice: → 1
Enter element: → 11

Enter the choice: → 1
Enter element: → 12

Enter the choice: → 4
12

Enter the choice: → 3
11
12

# Experiment -8

Aim Write a program to perform stack operation push, pop, peek and display. (Array)

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

int push (int [], int);
int pop (int);
void peek (int [], int);
void display (int [], int);

int main () {
        int x, stack [20], top = -1;
        while (1) {
                printf ("Enter the choice :\n");
                printf ("1) Push \n2) Pop \n3) Display \n4)
                        Peek \n5) Exit : ");
                scanf ("%d", &x);

                switch (x) {
                        case (1):
                                top = push (stack, top);
                                break;
                        case (2);
                                top = pop (top);
                                break;
                        case (3):
```

```
                display (stack, top);
                break;
        case (4) :
                peek (stack, top);
                break;
        case (5):
                exit (0);
                break;
        default :
                printf("Enter the right choice :");

            }
    }
    return 0;
}

int push (int stack [],int top){
    int x;
    if (top!=19){
            printf ("Enter element:");
            scanf("%d",&x);
            top ++;
            stack [top] = x;
    }
    else
        printf( "Overflow\n");
    return top;
}
```

```
int pop (int top){
        if (top==-1){
                printf("Underflow\n");
        else
                top--;
        return top;
}

void peek (int stack[], int top){
        if (top==-1){
                printf("Underflow\n");
        else
                printf("%d\n", stack[top]);
}

void display (int stack[], int top){
        int i;
        if (top==-1)
                printf("Underflow\n");
        else{
                for (i=0; i<=top; i++)
                        printf("%d\n", stack[i]);
        }
}
```

OUTPUT:

Enter infix expression:

→ $\boxed{(A+B)*(C+D)}$

AB+CD+*

# Experiment - 9

**Aim** Write a program to convert infix expression to postfix expression.

```
#include <stdio.h>
void push (char ~~str~~);
char pop();
void peek();
void display();
int precedence (char);
int stack [100];
int top = -1, n=100;

int main (){
    char infix [100], x;
    printf("Enter infix expression = \n");
    gets(infix);
    push('(');
    int i;
    for (i=0; infix [i] != '\0' ; i++) {
        if (infix [i] == '(') {
            push (infix (i));
            continue;
        }
        if (infix [i] == ')') {
            while (stack [top] != '(') {
                x = pop()
                printf("%c", x);
}
```

```
            pop();
            continue;
        }
        if (infix[i]!='*' && infix[i]!='/' && infix[i]!='+'
            && infix[i]!='-'){
            printf("%c", infix[i]);
            continue;
        }
        else {
            if (precedence(infix[i])>precedence(stack[top]))
            {   x=infix[i];
                push(x);
            }
            else {
                while((precedence(stack[top])>=
                    precedence(infix[i])))){
                    x=pop();
                    printf("%c",x);
                }
                push(infix[i]);
            }
        }
    }
    while (stack[top]!='(') {
        x=pop();
        printf("%c",x);
    }
    return 0;
}
```

```
int precedence (char operator)
{
    switch (operator)
    {
        case '*':
            return 4;
            break;
        case '/':
            return 4;
            break;
        case '+':
            return 3;
            break;
        case '-':
            return 3;
            break;
        default:
            return 0;
            break;
    }
}

void push (char ch){
    if (top > n-1)
        printf("\n Overflow\n");
    else {
        top++;
        stack [top] = ch;
    }
}
```

```
char pop (){
    if (top == -1)
        printf ("underflow\n");
    else {
        int temp = top;
        top = top - 1;
        return stack [temp];
    }
}
```

OUTPUT :

Enter a string → hello

olleh

# Experiment -10

Aim Write a program to reverse a string using stack.

```
#include <stdio.h>

void push (char);
void pop();

char stack [20];
char str [20];
top = -1;

int main () {
        printf("Enter a string:");
        gets (str);
        int i;
        for (i=0; str [i] != '0'; i++) {
                push ( str [i]);
        }
        pop();
        return 0;
}

void push (char x) {
        top ++;
        stack [top] = x;
}
```

```
void pop(){
    int i;
    for(i=top; i>=0; i++)
        printf("%c", stack[i]);
}
```

Experiment - 6

Aim: Write a program to check balanced bracket using stack

```
#include <stdio.h>

void push (char);
int pop();

char stack [20];
char str [20];
int top= -1;

int main (){
        int i, f=1;
        char x;
        printf (" Enter equation : ");
        gets (str);
        for (i=0; str [i]!='\0'; i++){
                if (str [i]=='('||str [i]=='{'||str [i]=='[') {
                        push (str [i]);
                }
                else if (str [i]==')') {
                        x=pop();
                        if (x=='{'||x=='[')
                                f=0;
                }
```

```
        else if (str[i]==']'){
                x=pop();
                if (x==='{'||x=='('){
                        f=0;
                }
        }
        else if (str[i]=='}'){
                x=pop();
                if (x=='('||x=='[')
                        f=0;
        }
}

if (top>0)
        printf("Unbalanced");
else if (f==1)
        printf("Balanced");
else
        printf("Unbalanced");
return0;
}

void push (char x){
        if (top!=19){
                top++;
                stack[top]=x;
        }
        else
                printf("Overflow");
}
```

```
int pop(){
        char x;
        x= stack [top];
        top--;
        return x;
}
```

Experiment - 7

Aim: Write a program to implement enqueue, dequeue, and display operation Queue data structure (Array).

```
#include <stdio.h>
#include <stdlib.h>
int f=-1, r=-1;
int queue [30];

void enqueue();
void dequeue();
void display();

int main(){
        int ch;
        while (1){
                printf ("Enter the operation you
want to perform\n1: Enqueue\n 2: Dequeue\n 3 :
Display\n 4: Exit : ");
                scanf ("%d", &ch);
                switch (ch){
                        case 1:
                                enqueue();
                                break;
                        case 2:
                                dequeue();
                                break;
```

```
            case 3:
                    display();
                    break;
            case 4:
                    exit(0);
            default:
                    printf("Wrong choice!");
        }
    }
    return 0;
}

void enqueue(){
    if (r==29)
        printf("Overflow");
    else {
        if (f== -1){
            f++, r++;
            printf("Enter element:\n");
            scanf("%d", &queue[f]);
        }
        else {
            r++;
            printf("Enter element:\n");
            scanf("%d", &queue[r]);
        }
    }
}
```

```
void dequeue (){
    if (f == -1)
        printf("Underflow\n");
    else {
        printf("%d will be dequeued\n", queue[f]);
        f++;
    }
    if (f>r)
        f = -1, r = -1;
}

void display (){
    int i;
    ~~int~~ if (f == -1)
        printf("Underflow\n");
    else {
        printf("The elements in queue are\n");
        for (i=f; i<=r; i++)
            printf("%d\n", queue[i]);
    }
}
```