1) Queue.

Name: Acheta. A. Tangade.

USN : 1RN19CS004

```
#include<stdio.h>
#include
#define max_size 5
void accept(int q[max_size],int *front ,int *rear)
{       int value;
        if (*rear != max_size-1)
        {       printf("enter the choice to be inserted \n");
                scanf("%d", &value);
                (*rear)++;
                q[*rear] = value;
                printf ("\nInserted = %d \n", value);
        }
        else
                printf("Queue is full \n");
}
void del(int q[max_size] ,int *front, int *rear)
{       int elem;
        if (*front <=*rear)
        {       printf ("Deleted : %d \n", q[*front]);
                elem = q[*front];
                (*front)++;
        }
        else
        {       printf("Queue is empty \n");
                (*front) = 0;
                (*rear) = -1;
        }
}
```

```
Void display (int Q[max-size], int front, int rear)
{       if (front > rear)
                printf ("Queue is empty \n");
        else
        {       int i;
                printf ("Queue ~~is empty~~ elements are : \n");
                for (i = front ; i <= rear ; i++)
                {       printf ("%d", Q[i]);
                }
        }
}
int main()
{       int Q[max-size], front = 0, rear = -1, ch;
        while (1)
        { printf ("enter the choice \n");
          printf ("1. accept \n 2. del \n 3. display \n 4. exit");
          scanf ("%d", &ch);
          switch (ch)
          {     case 1: accept (Q, front, &rear);
                        break;
                case 2: del (Q, &front, &rear);
                        break;
                case 3: display (Q, front, rear);
                        break;
                default: return 0;
          }
        }
}
```

(2) infix to postfix expression

```
#include<stdio.h>
#include<string.h>
int F(char symbol)
{
    switch(symbol)
    {   case '+':
        case '-': return 2;
        case '*':
        case '/': return 4;
        case '^':
        case '$': return 5;
        case '(': return 0;
        case '#': return -1;
        default : return 8;
    }
}
int G(char symbol)
{   switch(symbol)
    {   case '+':
        case '-': return 1;
        case '*':
        case '/': return 3;
        case '^':
        case '$': return 6;
        case '(': return 9;
        case ')': return 0;
        default: return 7;
    }
}
```

```
Void push(char s[20], int *top, char symbol)
{
    (*top)++;
    s[*top] = symbol;
}
Void pop(char s[20], int *top)
{
    char elem;
    elem = s[*top];
    (*top)--;
    return elem;
}
int main()
{
    char infix_expr[20];
    char s[20], symbol, postfix[20];
    int i, top = -1, j = 0;
    printf("enter the ~~postfix~~ infix expression \n");
    scanf("%s", &infix_expr);
    push(s, &top, '#');
    for(i=0; i<strlen(infix_expr); i++)
    {
        symbol = infix_expr[i];
        while(F(s[top]) > G(symbol))
            postfix[j++] = pop(s, &top);
        if(F(s[top]) != G(symbol))
            push(s, &top, symbol);
        else
            top--;
    }
    while(s[top] != '#')
        postfix[j++] = pop(s, &top);
    postfix[++j] = '\0';
    printf("postfix exp is %s", postfix);
    return 0;
}
```

③ Evaluation of postfix exp.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
void push (int s[20], int *top, int operand)
{
    (*top)++;
    s[*top] = operand;
}
int pop(int s[20], int *top)
{
    int elem;
    elem = s[*top];
    (*top)--;
    return elem;
}
int evaluate (int op1, char symbol, int op2)
{
    switch (symbol)
    {
        case '+' : return (op1 + op2);
        case '-' : return (op1 - op2);
        case '*' : return (op1 * op2);
        case '/' : return (op1 / op2);
        case '^' : return pow (op1, op2);
    }
}
```

```
int main()
{
    char postfix_expr[20];
    int s[20], op1, op2, res;
    char symbol;
    int i, top = -1;
    printf("enter postfix exp. \n");
    scanf("%s", &postfix_expr);
    for(i=0; i<strlen(postfix_expr); i++)
    {
        symbol = postfix_expr[i];
        if (isdigit(symbol)).
        {
            push(s, &top, symbol - '0');
        }
        else
        {
            op2 = pop(s, &top);
            op1 = pop(s, &top);
            res = evaluate(op1, symbol, op2);
            push(s, &top, res);
        }
    }
    printf("result = %d \n", pop(s, &top));
    return 0;
}
```