```c
//EXP01
#include <stdio.h>
#define SIZE 100
void displayArray(int arr[], int n)
{
    printf("Array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void insertElement(int arr[], int *n, int pos, int elem)
{
    if (*n == SIZE)
    {
        printf("Array is full, cannot insert element.\n");
        return;
    }
    if (pos < 0 || pos > *n)
    {
        printf("Invalid position.\n");
        return;
    }
    for (int i = *n; i > pos; i--)
    {
        arr[i] = arr[i - 1];
    }

    arr[pos] = elem;
    (*n)++;
}

void deleteElement(int arr[], int *n, int pos)
{
    if (*n == 0)
    {
        printf("Array is empty, cannot delete element.\n");
        return;
    }
    if (pos < 0 || pos >= *n)
    {
        printf("Invalid position.\n");
        return;
    }
    for (int i = pos; i < *n - 1; i++)
    {
        arr[i] = arr[i + 1];
    }
```

```
    (*n)--;
}

int main() {
    int arr[SIZE], n = 0, choice, elem, pos;
    arr[0] = 10; arr[1] = 20; arr[2] = 30; arr[3] = 40;
    n = 4;
    while (1) {
            printf("\nMenu:\n");
            printf("1. Insert Element\n");
            printf("2. Delete Element\n");
            printf("3. Display Array\n");
            printf("4. Exit\n");
            printf("Enter your choice: ");
            scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter element to insert: ");
                scanf("%d", &elem);
                printf("Enter position to insert (0 to %d): ", n);
                scanf("%d", &pos);
                insertElement(arr, &n, pos, elem);
                break;
            case 2:
                printf("Enter position to delete (0 to %d): ", n - 1);
                scanf("%d", &pos);
                deleteElement(arr, &n, pos);
                break;
            case 3:
                displayArray(arr, n);
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice.\n");
        }
    }
 return 0;
}

//EXP02

#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
void push(int);
void pop();
void display();
int stack[SIZE],top = -1;
```

```
void main()
{
    int value, choice;
    while(1){
            printf("\n\n*** MENU ***\n");
            printf("1. Push\n2. Pop\n3. Display\n4. Exit");
            printf("\nEnter your choice: ");
            scanf("%d",&choice);
            switch(choice)
            {
                case 1:
                    printf("Enter the value to be inserted: ");
                    scanf("%d",&value);
                    push(value);
                    break;
                case 2:
                    pop();
                    break;
                case 3:
                    display();
                    break;
                case 4:
                    exit(0);
                default:
                    printf("\nWrong selection!!! Try again!!!");
             }
            }
}
void push(int value)
{
    if(top == SIZE-1)
    printf("\nStack is Full!!! Insertion is not possible!!!");
    else{
            top++;
            stack[top] = value;
            printf("\nInsection success!!!");
        }
}
void pop()
{
    if(top == -1)
    printf("\nStack is empty!!! Deletion is not possible!!!");
    else{
            printf("\nDeleted : %d , stack[top]");
            top--;
        }
}
void display()
{
    if(top == -1)
```

```
    printf("\nStack is Empty!!!");
    else{
        int i;
        printf("\nStack element are:\n");
        for(i=top; i>=0; i--)
        printf("%d\n", stack[i]);
        }
}



//EXP03

#include<stdio.h>
#include<string.h>
#include<ctype.h>
char stack[100];
int top = -1;
void push(char x)
{
    stack[++top] = x ;
}
char pop()
{
    if(top == -1)
    return -1;
    else
    return stack[top--];
}
int priority(char x)
{
    if(x == '(')
    return 0;
    if(x == '+' || x == '-')
    return 1;
    if(x == '*' || x == '/')
    return 2;
    if(x == '^')
    return 3;
        return 0;
}
int main()
{
     char exp[100];
     char *e, x;
     printf("Enter the expression : ");
     scanf("%s",exp);
     printf("\n");
     e = exp;
     while(*e != '\0')
```

```
        {
           if(isalnum(*e))
           printf("%c ",*e);
           else if(*e == '(')
           push(*e);
           else if(*e == ')')
            {
               while((x = pop()) != '(')
               printf("%c ", x);
            }
           else
            {
               while(priority(stack[top]) >= priority(*e))
               printf("%c ",pop());
               push(*e);
            }
             e++;
        }
    while(top != -1)
    {
        printf("%c ",pop());
    }
    return 0;
}

//EXP04

#include<stdio.h>
#include<ctype.h>
#include<math.h>
int stack[50];
int top = -1;
int push(int x);
int pop();
void main()
{
    char exp[50];
    int i=0, op1, op2, result;
    printf("Enter the postfix expression: ");
    scanf("%s",exp);
    while(exp[i] != '\0')
    {
        if(isdigit(exp[i]))
        push(exp[i]-'0'); //for ascii char from 0 to 9 integer=char-'0'
        else
            {
                op1 = pop();
                op2 = pop();
                switch(exp[i])
                {
```

```
                    case '+':
                             push(op2+op1);
                             break;
                    case '-':
                            push(op2-op1);
                            break;
                    case '*':
                            push(op2*op1);
                            break;
                    case '/':
                            push(op2/op1);
                            break;
                    case '^':
                            push(pow(op2,op1));
                            break;
                }
            }
        i++;
    }
    result = pop();
    printf("\nThe answer of postfix expression %s = %d\n",exp,result);
}
int push(int x)
{
    top++;
    stack[top] = x;
}
int pop()
{
    int y;
    y = stack[top];
    top--;
    return y;
}

//EXP05

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define SIZE 10
void enQueue(int);
void deQueue();
void display();
int queue[SIZE], front = -1, rear = -1;
void main()
{
    int value, choice;
    while(1)
    {
```

```
        printf("\n\n***** MENU *****\n");
        printf("1. Insertion\n2. Deletion\n3. Display\n4. Exit");
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                    printf("Enter the value to be inserted: ");
                    scanf("%d",&value);
                    enQueue(value);
                    break;
            case 2:
                    deQueue();
                    break;
            case 3:
                    display();
                    break;
            case 4:
                    exit(0);
            default:
                    printf("\nWrong selection!!! Try again!!!");
                      }
    }
}
void enQueue(int value)
{
    if(rear == SIZE-1)
    printf("\nQueue is Full! Insertion is not possible!");
    else{
            if(front == -1)
            front = 0;
            rear++;
            queue[rear] = value;
            printf("\n Insertion success!!!");
        }
}
void deQueue()
{
    if(front > rear || (front ==-1 && rear ==-1))
    printf("\nQueue is Empty!!! Deletion is not possible!!!");
    else{
        printf("\nDeleted : %d", queue[front]);
        front++;
        if(front > rear)
        front = rear = -1;
        }
}
void display()
{
    if(front > rear || (front ==-1 && rear ==-1))
    printf("\nQueue is Empty!!!");
```

```
    else{
        int i;
        printf("\nQueue elements are:\n");
        for(i=front; i<=rear; i++)
        printf("%d\t",queue[i]);
    }
}

//EXP06

#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
void enQueue(int);
void deQueue();
void display();
int cQueue[SIZE],front=-1,rear=-1;
void main()
{
    int choice,value;
    while(1) {
    printf("\nSelect one\n");
    printf("1.Insert\n2.Delete\n3.Display\n4.Exit\n");
    printf("Enter your choice:");
    scanf("%d",&choice);
    switch(choice) {
    case 1:
        printf("Enter the value to be inserted");
        scanf("%d",&value);
        enQueue(value);
        break;
    case 2:
        deQueue();
        break;
    case 3:
        display();
        break;
    case 4:
        exit(0);
    default:
        printf("\n Wrong. Select the correct choice\n");
     }
  }
}
void enQueue(int value)
{
  if((front == 0 && rear ==-1)||(front == rear+1))
  printf("\nQueue if Full\n");
  else{
        if(rear == SIZE-1 && front != 0)
```

```
        rear=-1;
        cQueue[++rear]=value;
        printf("\nValue is Inserted\n");
        if(front ==-1)
        front = 0;
      }
}
void deQueue()
{
    if(front == -1 && rear == -1)
    printf("\n CIR Queue is Empty\n");
else{
    printf("\n Deleted element: %d\n",cQueue[front++]);
    if(front == SIZE)
    front = 0;
    if(front-1 == rear)
    front = rear = -1;
    }
}
void display()
{
    if(front == -1)
    printf("\nCIR Queue is Empty");
    else{
        int i = front;
        printf("\nCIR Queue Elements are :\n");
        if(front <= rear)
        {
            while(i <= rear)
            printf("%d\t",cQueue[i++]);
        }
        else{
            while(i <= SIZE - 1)
            printf("%d\t",cQueue[i++]);
            i = 0;
            while(i <=rear)
            printf("%d\t",cQueue[i++]);
            }
        }
}


//EXP07

#include<stdio.h>
#include<stdlib.h>
void insertAtBeginning(int);
void insertAtEnd(int);
void insertBetween(int,int,int);
void display();
```

```
void removeBeginning();
void removeEnd();
void removeSpecific(int);
struct Node
{
    int data;
    struct Node *next;
}
*head = NULL;
void main()
{
    int choice,value,choice1,loc1,loc2;
    while(1){
        printf("\nmainMenu:\n1.Insert\n2. Display\n3. Delete\n4. Exit\nEnter your
choice: ");
        scanf("%d",&choice);
        switch(choice)
         {
           case 1:
                printf("Enter the value to be inserted: ");
                scanf("%d",&value);
                while(1) {
                    printf("Location to insert: \n1. At Beginning\n2. At End\n3.
                    Between\nEnter your choice: ");
                    scanf("%d",&choice1);
                switch(choice1)
                   {
                    case 1:
                    insertAtBeginning(value);
                    break;

                    case 2:
                    insertAtEnd(value);
                    break;

                    case 3:
                        printf("Enter the two values between which value to be
inserted: ");
                        scanf("%d%d",&loc1,&loc2);
                        insertBetween(value,loc1,loc2);
                    break;

                    default:
                        printf("\nWrong Input Try again \n\n");
                        goto mainMenu;
                    }
                goto subMenuEnd;
                        }
               subMenuEnd:
               break;
```

```
            case 2:
                 display();
                 break;

            case 3:
                 printf("How do you want to Delete: \n1. From Beginning\n2. From
End\n3. Specific\nEnter your choice: ");
                 scanf("%d",&choice1);
                 switch(choice1)
                 {
                    case 1:
                    removeBeginning();
                    break;

                    case 2:
                    removeEnd();
                    break;

                    case 3:
                    printf("Enter the value which you wanto delete: ");
                    scanf("%d",&loc2);
                    removeSpecific(loc2);
                    break;

                    default: printf("\nWrong Input Try again\n\n");
                    goto mainMenu;
            }

            default:
            printf("\nWrong input Try again \n\n");
        }
    }
}
void insertAtBeginning(int value)
{
        struct Node *newNode;
        newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = value;
        if(head == NULL)
        {
            newNode->next = NULL;
            head = newNode;
        }
        else
        {
            newNode->next = head;
            head = newNode;
        }
        printf("\nOne node inserted\n");
```

```
}
void insertAtEnd(int value)
{
        struct Node *newNode;
        newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = value;
        newNode->next = NULL;
        if(head == NULL)
            head = newNode;
        else
        {
            struct Node *temp = head;
            while(temp->next != NULL)
            temp = temp->next;
            temp->next = newNode;
        }
        printf("\nOne node inserted!!!\n");
}
void insertBetween(int value, int loc1, int loc2)
{
        struct Node *newNode;
        newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = value;
        if(head == NULL)
        {
            newNode->next = NULL;
            head = newNode;
        }
        else
        {
            struct Node *temp = head;
            while(temp->data != loc1 && temp->data != loc2)
            temp = temp->next;
            newNode->next = temp->next;
            temp->next = newNode;
        }
        printf("\nOne node inserted\n");
}
void removeBeginning()
{
        if(head == NULL)
            printf("\n\nList is Empty!!!");
        else
        {
           struct Node *temp = head;
                if(head->next == NULL)
                      {
                       head = NULL;
                       free(temp);
                      }
```

```
                else
                {
                    head = temp->next;
                    free(temp);
                    printf("\nOne node deleted\n\n");
                }
        }
}
void removeEnd()
{
        if(head == NULL)
        {
         printf("\nList is Empty\n");
        }
        else
        {
            struct Node *temp1 = head,*temp2;
            if(head->next == NULL)
                    head = NULL;
             else
              {
                while(temp1->next != NULL)
                {
                temp2 = temp1;
                temp1 = temp1->next;
                 }
             temp2->next = NULL;
            }
                free(temp1);
                 printf("\nOne node deleted\n\n");
            }
   }
void removeSpecific(int delValue)
    {
        struct Node *temp1 = head, *temp2;
        while(temp1->data != delValue)
        {
            if(temp1 -> next == NULL)
                {
                printf("\nGiven node not found in the list");
                goto functionEnd;
                }
            temp2 = temp1;
            temp1 = temp1 -> next;
        }
        temp2 -> next = temp1 -> next;
        free(temp1);
        printf("\nOne node deleted\n\n");
        functionEnd:
        printf("\nEnd of function remove at specific");
```

```
        }
void display()
{
        if(head == NULL)
        {
            printf("\nList is Empty\n");
        }
        else
        {
            struct Node *temp = head;
            printf("\n\nList elements are - \n");
            while(temp->next != NULL)
                {
                    printf("%d --->",temp->data);
                    temp = temp->next;
                 }
            printf("%d --->NULL",temp->data);
        }
}
                .
```