1. write a c program to convert inflex to pstfix.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#define MAX 100
typedef struct {
   char arr[MAX];
   int top;
} Stack;
void initStack(Stack *s) {
   s->top = -1;
}
void push(Stack *s, char c) {
   if (s->top < (MAX - 1)) {
      s->arr[++(s->top)] = c;
   } else {
      printf("Stack overflow\n");
      exit(1);
   }
}
char pop(Stack *s) {
   if (s->top >= 0) {
      return s->arr[(s->top)--];
   } else {
      printf("Stack underflow\n");
      exit(1);
   }
}
char peek(Stack *s) {
   if (s->top >= 0) {
      return s->arr[s->top];
   } else {
      return '\0';
   }
}
int isOperator(char c) {
   return (c == '+' || c == '-' || c == '*' || c == '/');
}
int precedence(char op) {
   switch (op) {
      case '+':
      case '-':
         return 1;
      case '*':
      case '/':
         return 2;
      default:
         return 0;
   }
}
int main() {
   char infix[MAX], postfix[MAX];
   Stack s;
   int i, k = 0;
   initStack(&s);
   printf("Enter infix expression: ");
   fgets(infix, MAX, stdin);
   infix[strcspn(infix, "\n")] = '\0';
   for (i = 0; infix[i] != '\0'; i++) {
      char c = infix[i];
      if (isalpha(c) || isdigit(c)) {
         postfix[k++] = c;
      }
```

```
        else if (c == '(') {
            push(&s, c);
        }
        else if (c == ')') {
            while (peek(&s) != '(') {
                postfix[k++] = pop(&s);
            }
            pop(&s); // Remove '(' from stack
        }
        else if (isOperator(c)) {
            while (s.top != -1 && precedence(peek(&s)) >= precedence(c)) {
                postfix[k++] = pop(&s);
            }
            push(&s, c);
        }
    }
    while (s.top != -1) {
        postfix[k++] = pop(&s);
    }
    postfix[k] = '\0';
    printf("Postfix expression: %s\n", postfix);
    return 0;
}
```

Sample ouput:

```
Enter infix expression:A + B * (C - D) / E + B * (C - D) / E
Postfix expression: BCD-*E/+
```

2. writ a c program for array implementation using queue.

```
#include <stdio.h>
#define MAX 100
int main() {
    int queue[MAX];
    int front = -1, rear = -1;
    int item;
    printf("Enqueue 10\n");
    if (rear == MAX - 1) {
        printf("Queue is full!\n");
    } else {
        if (front == -1) {
            front = 0; // Queue was empty
        }
        rear++;
        queue[rear] = 10;
        printf("Enqueued 10\n");
    }
    printf("Enqueue 20\n");
    if (rear == MAX - 1) {
        printf("Queue is full!\n");
    } else {
        rear++;
        queue[rear] = 20;
        printf("Enqueued 20\n");
    }
    printf("Display queue:\n");
    if (front == -1) {
        printf("Queue is empty!\n");
    } else {
        printf("Queue elements: ");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
    printf("Dequeue\n");
```

```
    if (front == -1) {
        printf("Queue is empty!\n");
    } else {
        printf("Dequeued %d\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1; // Queue is empty
        }
    }
    printf("Display queue:\n");
    if (front == -1) {
        printf("Queue is empty!\n");
    } else {
        printf("Queue elements: ");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
    printf("Enqueue 30\n");
    if (rear == MAX - 1) {
        printf("Queue is full!\n");
    } else {
        rear++;
        queue[rear] = 30;
        printf("Enqueued 30\n");
    }
    printf("Display queue:\n");
    if (front == -1) {
        printf("Queue is empty!\n");
    } else {
        printf("Queue elements: ");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
    printf("Dequeue\n");
    if (front == -1) {
        printf("Queue is empty!\n");
    } else {
        printf("Dequeued %d\n", queue[front]);
        front++;
        if (front > rear) {
            front = rear = -1; // Queue is empty
        }
    }
    printf("Display queue:\n");
    if (front == -1) {
        printf("Queue is empty!\n");
    } else {
        printf("Queue elements: ");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
    printf("Exiting...\n");
    return 0;
}
```

Sample ouput:

```
Enqueue 10
Enqueued 10
Enqueue 20
```

```
Enqueued 20
Display queue:
Queue elements: 10 20
Dequeue
Dequeued 10
Display queue:
Queue elements: 20
Enqueue 30
Enqueued 30
Display queue:
Queue elements: 20 30
Dequeue
Dequeued 20
Display queue:
Queue elements: 30
Exiting…
```

3.writ a c program for linked list implementation using queue.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
int main() {
    Node* front = NULL; // Front of the queue
    Node* rear = NULL;  // Rear of the queue
    Node* temp;
    int item;
    printf("Enqueue 10\n");
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = 10;
    newNode->next = NULL;
    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    printf("Enqueue 20\n");
    newNode = (Node*)malloc(sizeof(Node));
    newNode->data = 20;
    newNode->next = NULL;
    rear->next = newNode;
    rear = newNode;
    printf("Display queue:\n");
    if (front == NULL) {
        printf("Queue is empty!\n");
    } else {
        temp = front;
        printf("Queue elements: ");
        while (temp != NULL) {
            printf("%d ", temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
    if (front == NULL) {
        printf("Queue is empty!\n");
    } else {
        temp = front;
        printf("Dequeued %d\n", front->data);
        front = front->next;
        if (front == NULL) {
```

```
        rear = NULL;
    }
    free(temp);
}
printf("Display queue:\n");
if (front == NULL) {
    printf("Queue is empty!\n");
} else {
    temp = front;
    printf("Queue elements: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
printf("Enqueue 30\n");
newNode = (Node*)malloc(sizeof(Node));
newNode->data = 30;
newNode->next = NULL;
if (rear == NULL) {
    front = rear = newNode;
} else {
    rear->next = newNode;
    rear = newNode;
}
printf("Display queue:\n");
if (front == NULL) {
    printf("Queue is empty!\n");
} else {
    temp = front;
    printf("Queue elements: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
if (front == NULL) {
    printf("Queue is empty!\n");
} else {
    temp = front;
    printf("Dequeued %d\n", front->data);
    front = front->next;
    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
}
printf("Display queue:\n");
if (front == NULL) {
    printf("Queue is empty!\n");
} else {
    temp = front;
    printf("Queue elements: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
while (front != NULL) {
    temp = front;
    front = front->next;
```

```
        free(temp);
    }
    printf("Exiting...\n");
    return 0;
}
```

Sample ouput:

```
Enqueue 10
Enqueue 20
Display queue:
Queue elements: 10 20
Dequeued 10
Display queue:
Queue elements: 20
Enqueue 30
Display queue:
Queue elements: 20 30
Dequeued 20
Display queue:
Queue elements: 30
Exiting...
```