# Task 1:

# Code:

#include<iostream>

using namespace std;

class Queue

{

int front, rear, size, arr[5];

public:

Queue()

{

front = rear = -1;

size = 5;

}

bool isfull()

{

if (rear + 1 % size == front)

{

return true;

}

return false;

}

void insert\_front(int x) {

if (front == -1 && rear == -1)

{

front = rear = 0;

arr[front] = x;

cout << "Inserted At Front " << endl;

}

else if (front == 0 && rear == 0)

{

front = size - 1;

arr[front] = x;

cout << "Inserted At Front " << endl;

}

else if (!isfull())

{

front--;

arr[front] = x;

cout << "Inserted At Front " << endl;

}

}

void insert\_last(int x)

{

if (front == -1 && rear == -1)

{

front = rear = 0;

arr[rear] = x;

cout << "Inserted At Last " << endl;

}

else if (front == 0 && rear == 0)

{

rear++;

arr[rear] = x;

cout << "Inserted At Last " << endl;

}

else if (!isfull())

{

rear++;

arr[rear] = x;

cout << "Inserted At Last " << endl;

}

}

void display()

{

for (int i = 0; i < 5; i++)

{

cout << arr[i] << " ";

}

cout << endl;

}

bool isempty()

{

if (front == -1 && rear == -1)

{

return true;

}

return false;

}

void Delete\_Front()

{

if (!isempty())

{

if (front == rear)

{

front = rear = -1;

cout << "Deleted From Front" << endl;

}

else if (front = size - 1)

{

arr[front] = -1;

front = 0;

cout << "Deleted From Front" << endl;

}

else {

arr[front] = -1;

front++;

cout << "Deleted From Front" << endl;

}

}

}

void Delete\_Last()

{

if (!isempty())

{

if (front == rear)

{

front = rear = -1;

cout << "Deleted From Last" << endl;

}

else if (rear == 0 && front != 0)

{

arr[rear] = -1;

rear = size - 1;

cout << "Deleted From Last" << endl;

}

else {

arr[rear] = -1;

rear--;

cout << "Deleted From Last" << endl;

}

}

}

int Get\_Front()

{

return arr[front];

}

int Get\_Last()

{

return arr[rear];

}

};

int main()

{

Queue obj;

obj.insert\_front(1);

obj.insert\_front(2);

obj.insert\_front(3);

obj.insert\_last(4);

obj.insert\_last(5);

obj.display();

obj.Delete\_Front();

obj.Delete\_Last();

obj.Delete\_Last();

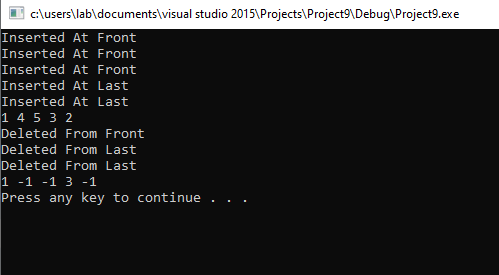
obj.display();

system("pause");

return 0;

}

# Output:



# Task 2:

# Code:

#include<iostream>

#include<string>

#include<stack>

using namespace std;

int operat(char);

string prefix(string x)

{

string rev = "";

stack <char> y;

int size = x.length();

char a = 'a', b, c;

for (int i = 0, j = size - 1; i < size; i++, j--)

{

a = x[j];

rev = rev + a;

}

x = "";

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

{

b = rev[i];

if ((b >= 65 && b <= 90) || (b >= 97 && b <= 122))

{

x = x + b;

}

else if (y.empty() == true)

{

y.push(b);

}

else if (y.empty() != true)

{

c = y.top();

while ((y.empty() != true) && (operat(y.top()) <= operat(b)))

{

x = x + y.top();

y.pop();

}

y.push(b);

}

}

while (y.empty() != true)

{

x = x + y.top();

y.pop();

}

rev = "";

for (int i = 0, j = size - 1; i < size; i++, j--)

{

a = x[j];

rev = rev + a;

}

return rev;

}

int main()

{

string user;

cout << "Enter user : ";

getline(cin, user);

user = prefix(user);

cout << user << endl;

system("pause");

return 0;

}

int operat(char x)

{

if (x == '\*' || x == '/')

{

return 1;

}

else if (x == '+' || x == '-')

{

return 2;

}

# }

# Output:

