**INPUT CODE:**

#include<iostream>

using namespace std;

#define SIZE 5

class dequeue {

int a[10], front, rear, count;

public:

dequeue();

void add\_at\_beg(int);

void add\_at\_end(int);

void delete\_fr\_front();

void delete\_fr\_rear();

void display();

};

dequeue::dequeue() {

front = -1;

rear = -1;

count = 0;

}

void dequeue::add\_at\_beg(int item) {

int i;

if (front == -1) {

front++;

rear++;

a[rear] = item;

count++;

} else if (rear >= SIZE - 1) {

cout << "\nInsertion is not possible, overflow!!!!";

} else {

for (i = count; i >= 0; i--) {

a[i] = a[i - 1];

}

a[i] = item;

count++;

rear++;

}

}

void dequeue::add\_at\_end(int item) {

if (front == -1) {

front++;

rear++;

a[rear] = item;

count++;

} else if (rear >= SIZE - 1) {

cout << "\nInsertion is not possible, overflow!!!";

return;

} else {

a[++rear] = item;

}

}

void dequeue::display() {

if (front == -1) {

cout << "Dequeue is empty.";

return;

}

for (int i = front; i <= rear; i++) {

cout << a[i] << " ";

}

}

void dequeue::delete\_fr\_front() {

if (front == -1) {

cout << "Deletion is not possible:: Dequeue is empty";

return;

} else {

if (front == rear) {

front = rear = -1;

return;

}

cout << "The deleted element is " << a[front];

front = front + 1;

}

}

void dequeue::delete\_fr\_rear() {

if (front == -1) {

cout << "Deletion is not possible: Dequeue is empty";

return;

} else {

if (front == rear) {

front = rear = -1;

}

cout << "The deleted element is " << a[rear];

rear = rear - 1;

}

}

int main() {

int c, item;

dequeue d1;

do {

cout << "\n\n\*\*\*\*DEQUEUE OPERATION\*\*\*\*\n";

cout << "\n1-Insert at beginning";

cout << "\n2-Insert at end";

cout << "\n3-Display";

cout << "\n4-Deletion from front";

cout << "\n5-Deletion from rear";

cout << "\n6-Exit";

cout << "\nEnter your choice<1-6>:";

cin >> c;

switch (c) {

case 1:

cout << "Enter the element to be inserted:";

cin >> item;

d1.add\_at\_beg(item);

break;

case 2:

cout << "Enter the element to be inserted:";

cin >> item;

d1.add\_at\_end(item);

break;

case 3:

d1.display();

break;

case 4:

d1.delete\_fr\_front();

break;

case 5:

d1.delete\_fr\_rear();

break;

case 6:

exit(1);

break;

default:

cout << "Invalid choice";

break;

}

} while (c != 7);

return 0;

}

--------------------------------------------------------------------------------------------------------------------------------------

**OUTPUT:**

\*\*\*\*DEQUEUE OPERATION\*\*\*\*

1-Insert at beginning

2-Insert at end

3-Display

4-Deletion from front

5-Deletion from rear

6-Exit

Enter your choice<1-6>:1

Enter the element to be inserted:23

\*\*\*\*DEQUEUE OPERATION\*\*\*\*

1-Insert at beginning

2-Insert at end

3-Display

4-Deletion from front

5-Deletion from rear

6-Exit

Enter your choice<1-6>:2

Enter the element to be inserted:34

\*\*\*\*DEQUEUE OPERATION\*\*\*\*

1-Insert at beginning

2-Insert at end

3-Display

4-Deletion from front

5-Deletion from rear

6-Exit

Enter your choice<1-6>:3

23 34

\*\*\*\*DEQUEUE OPERATION\*\*\*\*

1-Insert at beginning

2-Insert at end

3-Display

4-Deletion from front

5-Deletion from rear

6-Exit

Enter your choice<1-6>:4

The deleted element is 23

\*\*\*\*DEQUEUE OPERATION\*\*\*\*

1-Insert at beginning

2-Insert at end

3-Display

4-Deletion from front

5-Deletion from rear

6-Exit

Enter your choice<1-6>:5

The deleted element is 0

\*\*\*\*DEQUEUE OPERATION\*\*\*\*

1-Insert at beginning

2-Insert at end

3-Display

4-Deletion from front

5-Deletion from rear

6-Exit

Enter your choice<1-6>:6

=== Code Exited With Errors ===