DSL Assignment 8

Code:

```
#include <iostream>

using namespace std;

struct node

{

   int x;

   node *next, *prev;

};

class binary

{

   node *head = NULL, *temp = NULL, *head1 = NULL, *temp1 = NULL, *head2 =
NULL, *temp2 = NULL, *temp3 = NULL, *head3 = NULL;

   int c, i;


public:

   node *create();

   void insert();

   void binary1();

   void binary2();

   void add();

   void com();

   void comp();

   void display();

};

node *binary::create()

{

   node *p = new (struct node);

   cout << "Enter binary number bit by bit: ";
```

```
    cin >> c;
    p->x = c;
    p->next = NULL;
    p->prev = NULL;
    return p;
}
void binary::insert()
{
    node *p = create();

    if (head == NULL)
    {
        head = p;
    }
    else
    {
        temp = head;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = p;
        p->prev = temp;
    }
}
void binary::binary1()
{
    int a;
    cout << "Enter the number of bits of first number: ";
    cin >> a;
    head = NULL;
    for (i = 0; i < a; i++)
    {
        insert();
    }
    head1 = head;
    display();
    head = NULL;
    temp1 = head1;
}
void binary::binary2()
{
    int a;
    cout << "Enter the number of bits of second number: ";
    cin >> a;
    head = NULL;
    for (i = 0; i < a; i++)
    {
        insert();
    }
    head2 = head;
    display();
    head = NULL;
}

void binary::add()
{
    int carry = 0;
```

```
temp1 = head1;
while (temp1->next != NULL)
{
   temp1 = temp1->next;
}

temp2 = head2;
while (temp2->next != NULL)
{
   temp2 = temp2->next;
}

while (temp1 != NULL)
{
   node *p = new (struct node);
   p->next = NULL;
   p->prev = NULL;
   if (temp1->x == 0 && temp2->x == 0)
   {
      p->x = 0 + carry;
      carry = 0;
   }
   if (temp1->x == 0 && temp2->x == 1)
   {
      if (carry == 0)
      {
         p->x = 1;
         carry = 0;
      }
```

```
    else
    {
        p->x = 0;
        carry = 1;
    }
}
if (temp1->x == 1 && temp2->x == 0)
{
    if (carry == 0)
    {
        p->x = 1;
        carry = 0;
    }
    else
    {
        p->x = 0;
        carry = 1;
    }
}
if (temp1->x == 1 && temp2->x == 1)
{
    if (carry == 0)
    {
        p->x = 0;
        carry = 1;
    }
    else
    {
        p->x = 1;
```

```
            carry = 1;
        }
    }
    if (temp3 == NULL)
    {
        temp3 = p;
    }
    else
    {
        p->next = temp3;
        temp3 = p;
    }

    temp1 = temp1->prev;
    temp2 = temp2->prev;
}
node *p = new (struct node);
p->x = carry;
p->next = NULL;
p->prev = NULL;
if (temp3 == NULL)
{
    temp3 = p;
}
else
{
    p->next = temp3;
    temp3 = p;
}
```

```
    head3 = temp3;
    temp3 = head3;
    cout << "\n";
    while (temp3->next != NULL)
    {
        cout << " " << temp3->x;
        ;
        temp3 = temp3->next;
    }
    cout << " " << temp3->x << "\n";
}
void binary::com()
{
    while (temp1 != NULL)
    {
        node *p = new (struct node);
        p->next = NULL;
        p->prev = NULL;
        if (temp1->x == 0)
            p->x = 1;
        else
            p->x = 0;

        if (head == NULL)
        {
            head = p;
        }
        else
        {
```

```
        temp = head;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = p;
        p->prev = temp;
    }
    temp1 = temp1->next;
}
cout << "\n1's compliment of  Binary number is: ";
display();

int f = 0;
while (temp != NULL)
{
    if (temp->x == 1)
    {
        temp->x = 0;
    }
    else
    {
        temp->x = 1;
        f = 1;
        break;
    }
    temp = temp->prev;
}
if (f == 0)
```

```
    {
        node *p = new (struct node);
        p->next = NULL;
        p->prev = NULL;
        p->x = 1;
        temp = head;
        head = p;
        head->next = temp;
        temp->prev = head;
    }
    cout << "\n2's compliment of  binary number is: ";
    display();
}
void binary::comp()
{

    cout << "\nResult of First Binary Number: ";
    temp1 = head1;
    com();
    head = NULL;
    cout << "\nResult of Second Binary Number: ";
    temp1 = head2;
    com();
}

void binary::display()
{
    temp = head;
    cout << "\n";
    while (temp->next != NULL)

    {

        cout << " " << temp->x;

        ;

        temp = temp->next;

    }

    cout << " " << temp->x << "\n";

}

int main()

{

    binary b;

    int m;

    char ch;

    do

    {

        cout << "\nEnter the choice: ";

        cout << "\n1.Insert first binary number";

        cout << "\n2.insert second binary number";

        cout << "\n3.Add binary numbers";

        cout << "\n4.1's and 2's compliment of binary numbers";

        cout << "\n\n: ";

        cin >> m;

        switch (m)

        {

        case 1:

            b.binary1();

            break;

        case 2:

            b.binary2();
            break;
        case 3:
            cout << "The addition of binary numbers is: ";
            b.add();
            break;
        case 4:
            b.comp();
            break;
        default:
            cout << "unknown choice";
        }
        cout << " \nPress 'y' to continue: ";
        cin >> ch;
    } while (ch == 'y' || ch == 'Y');

    return 0;
}
```

<u>Output:</u>

```
PS D:\Codes\DSL> cd "d:\Codes\DSL\" ; if ($?) { g++ a8_final.cpp -o a8_final } ; if ($?)
{ .\a8_final }

Enter the choice:
1.Insert first binary number
2.insert second binary number
3.Add binary numbers
4.1's and 2's compliment of binary numbers
: 1
```

```
Enter the number of bits of first number: 3

Enter binary number bit by bit: 1

Enter binary number bit by bit: 0

Enter binary number bit by bit: 1

1 0 1

Press 'y' to continue: y


Enter the choice:

1.Insert first binary number

2.insert second binary number

3.Add binary numbers

4.1's and 2's compliment of binary numbers

: 2


Enter the number of bits of second number: 3

Enter binary number bit by bit: 1

Enter binary number bit by bit: 0

Enter binary number bit by bit: 1

1 0 1

Press 'y' to continue: y


Enter the choice:

1.Insert first binary number

2.insert second binary number

3.Add binary numbers

4.1's and 2's compliment of binary numbers

: 3
```

```
The addition of binary numbers is:
 1 0 1 0
Press 'y' to continue: y

Enter the choice:
1.Insert first binary number
2.insert second binary number
3.Add binary numbers
4.1's and 2's compliment of binary numbers
: 4

Result of First Binary Number:
1's compliment of  Binary number is:
 0 1 0

2's compliment of  binary number is:
 0 1 1

Result of Second Binary Number:
1's compliment of  Binary number is:
 0 1 0

2's compliment of  binary number is:
 0 1 1

Press 'y' to continue: n
```

DSL Assignment 9

Code:

```
#include <iostream>

#include <string.h>

#define max 50

using namespace std;


class STACK

{

private:

        char a[max];

        int top;


public:

        STACK()

        {

                top = -1;

        }


        void push(char);

        void reverse();

        void convert(char[]);

        void palindrome();

};


void STACK::push(char c)

{

        top++;

        a[top] = c;
        a[top + 1] = '\0';

        cout << endl
                << c << " is pushed on stack.";
}

void STACK::reverse()
{
        char str[max];

        cout << "\n\nReverse string is: ";

        for (int i = top, j = 0; i >= 0; i--, j++)
        {
                cout << a[i];
                str[j] = a[i];
        }

        cout << endl;
}

void STACK::convert(char str[])
{
        int j, k, len = strlen(str);

        for (j = 0, k = 0; j < len; j++)
        {
                if (((int)str[j] >= 97 && (int)str[j] <= 122) || ((int)str[j] >= 65 &&
(int)str[j] <= 90))
                {
                        if ((int)str[j] <= 90)
                        {
                                str[k] = (char)((int)str[j] + 32);
                        }
                        else
                        {
                                str[k] = str[j];
                        }

                        k++;
                }
        }
        str[k] = ‘\0’;

        cout << endl
                 << “Converted String: “ << str << “\n”;
}

void STACK::palindrome()
{
        char str[max];
        int i, j;

        for (i = top, j = 0; i >= 0; i--, j++)
        {
                str[j] = a[i];
        }
        str[j] = ‘\0’;
        if (strcmp(str, a) == 0)
                cout << "\n\nString is a palindrome.";
        else
                cout << "\n\nString is not a palindrome.";
}

int main()
{
        STACK stack;

        char str[max];
        int i = 0;

        cout << "\nEnter string to be reversed and check is it palindrome or not: ";

        cin.getline(str, 50);

        stack.convert(str);

        while (str[i] != '\0')
        {
                stack.push(str[i]);
                i++;
        }

        stack.palindrome();

        stack.reverse();
}
```

Output:

```
PS D:\Codes\DSL> cd "d:\Codes\DSL\" ; if ($?) { g++ DSL_Ass9.cpp -o DSL_Ass9 } ; if
($?) { .\DSL_Ass9 }


Enter string to be reversed and check is it palindrome or not: "Poor Dan is in a
droop"


Converted String: poordanisinadroop


p is pushed on stack.

o is pushed on stack.

o is pushed on stack.

r is pushed on stack.

d is pushed on stack.

a is pushed on stack.

n is pushed on stack.

i is pushed on stack.

s is pushed on stack.

i is pushed on stack.

n is pushed on stack.

a is pushed on stack.

d is pushed on stack.

r is pushed on stack.

o is pushed on stack.

o is pushed on stack.

p is pushed on stack.


String is a palindrome.


Reverse string is: poordanisinadroop
```

DSL Assignment 10

Code:

```
#include <iostream>
using namespace std;
#define MAX 50
class stack
{
   char st[MAX], in[20], po[20];
   int top, k;

public:
   stack()
   {
      top = -1;
      k = 0;
   }
   void infixToPostfix();
   void evaluate();

private:
   void push(char);
   char pop();
   int precedence(char);
};

void stack::push(char ch)
{
   if (top == st[MAX])
   {
        cout << "Stack overflow" << endl;
    }
    else
    {
        top++;
        st[top] = ch;
    }
}

char stack::pop()
{
    if (top == -1)
    {
        cout << "Stack underflow" << endl;
        return 0;
    }
    else
    {
        int m = st[top];
        top--;
        return m;
    }
}

int stack::precedence(char ch)
{
    if (ch == '+' || ch == '-')
    {
        return 1;
    }
    else if (ch == '*' || ch == '/')
    {
        return 2;
    }
    else if (ch=='(')
    return 0;

}

void stack::infixToPostfix()
{
    int m;
    char left = '(', right = ')';
    cout << "Enter infix expression" << endl;
    cin >> in;
    for (int i = 0; in[i] != '\0'; i++)
    {
        if (isalpha(in[i]) == 1 || isdigit(in[i] == 1))
        {
            po[k] = in[i];
            k++;
        }
        else if (in[i] == left)
        {
            push(left);
        }
        else if (in[i] == right)
        {
            while ((m = pop()) != left)
            {
                po[k] = m;
                k++;
            }
        }
        else
        {
            while (precedence(st[top]) >= precedence(in[i]))
            {
                int m = pop();
                po[k] = m;
                k++;
            }
            push(in[i]);
        }
    }
    while (top >= 0)
    {
        po[k] = pop();
        k++;
    }
    po[k] = '\0';
    cout << "The postfix expression is" << endl;
    cout << po;
}

void stack::evaluate()
{
```

```
cout << "The postfix expression is" << endl
    << po << endl;
;
int a, b, res, temp;

top = -1;
for (int i = 0; po[i] != '\0'; i++)
{
   if (isdigit(po[i]) == 1)
   {
      push(po[i] - '0');
   }
   else
   {
      a = pop();
      b = pop();
      switch (po[i])
      {
      case '+':
         res = b + a;
         break;
      case '-':
         res = b - a;
         break;
      case '*':
         res = b * a;
         break;
      case '/':
         res = b / a;
```

```
            break;
          }
          push(res);
        }
      }
      temp = pop();
      cout << "The answer is " << temp << endl;
    }

    int main()
    {
      stack s;
      int op;
      do
      {
        cout << "\n=============== MENU =================" << endl;
        cout << "1 Convert Infix to Postfix and evaluate Postfix" << endl;
        cout << "2 Exit" << endl;
        cout << "=======================================" << endl;
        cin >> op;
        switch (op)
        {
        case 1:
          s.infixToPostfix();
          s.evaluate();

        default:
          cout << "Enter correct option" << endl;
        }
      } while (op != 1);
      return 0;
    }
```

Output:

```
PS D:\Codes\DSL sequentially> cd "d:\Codes\DSL sequentially\" ; if ($?) { g++ a10.cpp
-o a10 } ; if ($?) { .\a10 }


=============== MENU ==========================
1 Convert Infix to Postfix and evaluate Postfix
2 Exit
===============================================


Please enter your choice: 1


Enter Infix expression: ((1+2)*(4-2)/2)


===============================================


The postfix expression is: 12+42-*2/


The answer is: 3
===============================================
```

DSL Assignment 11

Code:

```
#include <iostream>
#define MAX 20
using namespace std;

class Queue
{

private:
   int job[MAX];
   int front, rear;

public:
   Queue()
   {
      front = rear = -1;
   }

   int isEmpty();
   int isFull();
   void enqueue(int);
   int delqueue();
   void display();
};
int Queue::isEmpty()
{
   return (front == rear) ? 1 : 0;
}
```

```
int Queue::isFull()
{
    return (rear == MAX - 1) ? 1 : 0;
}
void Queue::enqueue(int x)
{
    job[++rear] = x;
}
int Queue::delqueue()
{
    return job[++front];
}
void Queue::display()
{
    int i;
    cout << "\n";
    for (i = front + 1; i <= rear; i++)
        cout << job[i] << " ";
}
int main()
{
    Queue obj;
    int ch, x;
    do
    {
        cout << "\n 1.Insert Job\n 2.Delete Job\n 3.Display\n 4.Exit\n Enter your choice: 
";
        cin >> ch;
        switch (ch)
        {
```

```
case 1:
   if (!obj.isFull())
   {
      cout << "\n Enter data : ";
      cin >> x;
      obj.enqueue(x);
      cout << endl;
   }
   else
      cout << "Queue is overflow!!!\n\n";
   break;
case 2:
   if (!obj.isEmpty())
      cout << "\n Deleted Element = " << obj.delqueue() << endl;
   else
   {
      cout << "\n Queue is underflow!!!\n\n";
   }
   cout << "\nRemaining Jobs : \n";
   obj.display();
   break;
case 3:
   if (!obj.isEmpty())
   {
      cout << "\n Queue contains : \n";
      obj.display();
   }
   else
      cout << "\n Queue is empty!!!\n\n";
```

```
            break;
        case 4:
            cout << "\n Exiting Program.....";
        }
    } while (ch != 4);
    return 0;
}
```

Output:

```
PS D:\Codes\DSL> cd "d:\Codes\DSL\" ; if ($?) { g++ a11.cpp -o a11 } ; if ($?) { .\a11 }
 1.Insert Job
 2.Delete Job
 3.Display
 4.Exit
 Enter your choice: 1

 Enter data : 101

 1.Insert Job
 2.Delete Job
 3.Display
 4.Exit
 Enter your choice: 1
Enter data : 102
 1.Insert Job
 2.Delete Job
 3.Display
 4.Exit
 Enter your choice: 3
 Queue contains :
 101 102

 1.Insert Job
 2.Delete Job
 3.Display
 4.Exit
 Enter your choice: 2

Deleted Element = 101

Remaining Jobs :
 102

 1.Insert Job
 2.Delete Job
 3.Display
 4.Exit
 Enter your choice: 3

 Queue contains :
 102
 1.Insert Job
 2.Delete Job
 3.Display
 4.Exit
 Enter your choice: 4
 Exiting Program.....
```

DSL Assignment 12

Code:

```
#include <iostream>
using namespace std;
#define SIZE 10
class dequeue
{
   int a[20], f, r;

public:
   dequeue();
   void insert_at_beg(int);
   void insert_at_end(int);
   void delete_fr_front();
   void delete_fr_rear();
   void show();
};
dequeue::dequeue()
{
   f = -1;
   r = -1;
}
void dequeue::insert_at_end(int i)
{
   if (r >= SIZE - 1)
   {
      cout << "\nInsertion is not possible, overflow!!!";
   }
   else
```

```
    {
        if (f == -1)
        {
            f++;
            r++;
        }
        else
        {
            r = r + 1;
        }
        a[r] = i;
        cout << "\nInserted element is: " << a[r];
    }
}
void dequeue::insert_at_beg(int i)
{
    if (f == -1)
    {
        f = 0;
        a[++r] = i;
        cout << "\nInserted element is: " << i;
    }
    else if (f != 0)
    {
        a[--f] = i;
        cout << "\nInserted element is: " << i;
    }
    else
    {
        cout << "\nInsertion is not possible, overflow!!!";
    }
}
void dequeue::delete_fr_front()
{
    if (f == -1)
    {
        cout << "Deletion is not possible as dequeue is empty!";
        return;
    }
    else
    {
        cout << "The deleted element is: " << a[f];
        if (f == r)
        {
            f = r = -1;
            return;
        }
        else
            f = f + 1;
    }
}
void dequeue::delete_fr_rear()
{
    if (f == -1)
    {
        cout << "Deletion is not possible as dequeue is empty.";
        return;
    }
    else
    {
        cout << "The deleted element is: " << a[r];
        if (f == r)
        {
            f = r = -1;
        }
        else
            r = r - 1;
    }
}
void dequeue::show()
{
    if (f == -1)
    {
        cout << "Dequeue is empty";
    }
    else
    {
        for (int i = f; i <= r; i++)
        {
            cout << a[i] << " ";
        }
    }
}
int main()
{
    int c, i;
    dequeue d;
```

```
do
{
    cout << "\n\n======= Menu =========";
    cout << "\n1.Insert at beginning";
    cout << "\n2.Insert at end";
    cout << "\n3.Display Queue";
    cout << "\n4.Deletion from front";
    cout << "\n5.Deletion from rear";
    cout << "\n6.Exit";
    cout << "\nEnter your choice: ";
    cin >> c;
    switch (c)
    {
    case 1:
        cout << "Enter the element to be inserted: ";
        cin >> i;
        d.insert_at_beg(i);
        break;
    case 2:
        cout << "Enter the element to be inserted: ";
        cin >> i;
        d.insert_at_end(i);
        break;
    case 3:
        d.show();
        break;
    case 4:
        d.delete_fr_front();
```

```
            break;
        case 5:
            d.delete_fr_rear();
            break;
        case 6:
            exit(1);
            break;
        default:
            cout << "Invalid choice :(";
            break;
        }
    } while (c != 7);
}
```

Output:

```
PS D:\Codes\DSL> cd "d:\Codes\DSL\" ; if ($?) { g++ DSL_A12.cpp -o DSL_A12 } ; if ($?) { .\DSL_A12 }


======= Menu =========
1.Insert at beginning
2.Insert at end
3.Display Queue
4.Deletion from front
5.Deletion from rear
6.Exit
Enter your choice: 1
Enter the element to be inserted: 101
Inserted element is: 101
```

```
======= Menu =========

1.Insert at beginning

2.Insert at end

3.Display Queue

4.Deletion from front

5.Deletion from rear

6.Exit

Enter your choice: 2


Enter the element to be inserted: 303

Inserted element is: 303


======= Menu =========

1.Insert at beginning

2.Insert at end

3.Display Queue

4.Deletion from front

5.Deletion from rear

6.Exit

Enter your choice: 2


Enter the element to be inserted: 222

Inserted element is: 222


======= Menu =========

1.Insert at beginning

2.Insert at end

3.Display Queue

4.Deletion from front
```

```
5.Deletion from rear

6.Exit


Enter your choice: 3

101 303 222


======= Menu =========

1.Insert at beginning

2.Insert at end

3.Display Queue

4.Deletion from front

5.Deletion from rear

6.Exit

Enter your choice: 4


The deleted element is: 101


======= Menu =========

1.Insert at beginning

2.Insert at end

3.Display Queue

4.Deletion from front

5.Deletion from rear

6.Exit

Enter your choice: 5

The deleted element is: 222
```

```
======= Menu =========

1.Insert at beginning

2.Insert at end

3.Display Queue

4.Deletion from front

5.Deletion from rear

6.Exit

Enter your choice: 3

303


======= Menu =========

1.Insert at beginning

2.Insert at end

3.Display Queue

4.Deletion from front

5.Deletion from rear

6.Exit


Enter your choice: 6
```

Assignment 13

Code:

```
#include <iostream>
using namespace std;
const int MAX = 10;
class Order
{

   int quantity=1,pizza_code,total;
   string pizza_name;

public:

   void acceptOrder()
   {

      cout << "\nEnter Pizza Code ::";
      cin >> pizza_code;

      cout << "\nEnter quantity: ";
      cin >> quantity;
   }
   void calculateOrder()
   {
      if(pizza_code==1){
         pizza_name="Paneer_Tandoor";
         total=quantity*2;
```

```
        }
        if(pizza_code==2)
        {
        pizza_name="Pepperoni_Pizza";
            total=quantity*4;
        }
        if(pizza_code==3){
         pizza_name="Margherita_Pizza";
            total=quantity*10;
        }



    }
    void displayOrder()
    {
    calculateOrder();
        cout << "\n" <<pizza_code<<"\t\t"<< pizza_name << "\t   " <<quantity<< "\t
"<<total;


    }
    friend class Queue;
};


class Queue
{
    Order data[MAX];
    int front, rear;


public:
    Queue()
    {
        front = rear = -1;
    }
    void enqueue();
    void dequeue();
    int isFull();
    int isEmpty();
    void display();
};

int Queue::isFull()
{
    if ((front == 0 && rear == MAX - 1) || front == rear + 1)
        return 1;
    else
        return 0;
}

int Queue::isEmpty()
{
    if (front == -1 && rear == -1)
        return 1;
    else
        return 0;
}
//--------------------------------------------------
void Queue::enqueue()
{
    if (isFull())
    {
        cout << "\nCan't place order ! Queue is Full !";
    }
    else
    {
        Order temp;
        temp.acceptOrder();
        if (rear == MAX - 1 && front != 0)
        {
            rear = -1;
        }
        data[++rear] = temp;
        cout << "\nOrder Placed successfully";
        if (front == -1)
            front = 0;
    }
}
//--------------------------------------------------
void Queue::dequeue()
{
    if (isEmpty())
    {
        cout << "\nNo orders to Serve !";
    }
    else
    {
        front++;
        cout << "\nOrder Served successfully !";
        if (front == MAX)
```

```
            front = 0;
        if (front - 1 == rear)
            front = rear = -1;
    }
}

void Queue::display()
{
    if (isEmpty())
    {
        cout << "\nNo orders to display !";
    }
    else
    {
        int i = front;
        cout << "\n\t----------- Orders in Queue -------------";
        cout << "\n\nPizza ID \t Order Name \tQuantity \t Total";
        if (front <= rear)
        {
            while (i <= rear)
            {
                data[i].displayOrder();
                i++;
            }
        }
        else
        {
            while (i < MAX)
            {
                data[i].displayOrder();
                i++;
            }
            i = 0;
            while (i <= rear)
            {
                data[i].displayOrder();
                i++;
            }
        }
    }
}
//-----------------------------------------------------
int main()
{
    int ch;
    Queue q;
    cout << " Available Pizzas \t Price ";
    cout << "\n1. Paneer Tandoor \t $2 ";
    cout << "\n2. Pepperoni Pizza \t $4";
    cout << "\n3. Margherita Pizza \t $10";
    cout << "\n-------------- Pizza Parlor Menu --------------";
    cout << "\n1. Order";
    cout << "\n2. Serve Order";
    cout << "\n3. Display Orders";
    cout << "\n4. Exit";
    do
    {
        cout << "\n--------------------------------------";
        cout << "\nEnter your choice :: ";
        cin >> ch;
        cout << "\n--------------------------------------";
        switch (ch)
        {
        case 1:
            q.enqueue();
            break;
        case 2:
            q.dequeue();
            break;
        case 3:
            q.display();
            break;
        }
    } while (ch != 4);
}
```

<u>Output:</u>

```
Available Pizzas        Price

1. Paneer Tandoor        $2

2. Pepperoni Pizza       $4

3. Margherita Pizza      $10


-------------- Pizza Parlor Menu --------------

1. Order

2. Serve Order

3. Display Orders

4. Exit

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

Enter your choice :: 1


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

Enter Pizza Code ::3


Enter quantity: 10


Order Placed successfully

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

Enter your choice :: 1


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

Enter Pizza Code ::2


Enter quantity: 7


Order Placed successfully

---------------------------------------
```

```
Enter your choice :: 1


-------------------------------------
Enter Pizza Code ::1
Enter quantity: 8
Order Placed successfully
-------------------------------------
Enter your choice :: 3


          -------------------------------------
      ----------- Orders in Queue -------------


Pizza ID       Order Name     Quantity       Total
3            Margherita_Pizza       10           100
2            Pepperoni_Pizza 7      28
1            Paneer_Tandoor  8    16
-------------------------------------
Enter your choice :: 2
-------------------------------------
Order Served successfully !
-------------------------------------
Enter your choice :: 3
          -------------------------------------
      ----------- Orders in Queue -------------


Pizza ID       Order Name     Quantity       Total
2            Pepperoni_Pizza 7      28
1            Paneer_Tandoor  8    16
```