Roll No: BCSF18A004
Name: Irha Naveed
Title: Online Quiz 2

Challenge-A:

Part –A:

Time = O(N); if(x<y) then $O(N^2)$
Space = $O(N^2)$

Part-C:

Case-A false
Case-B true
Case-C false
Case-D false

Challenge-B:

```
int power(int x, int exp)
{
	int ans = x;
	for (int i = 1;i < exp;i++)
	{
		ans = ans*x;
	}
	return ans;
}

double evaluatePolynomial(double coefficients[], int exponents[], int N, int x)
{
	double ans = 0;
	for (int i = 0;i < N;i++)
	{
		ans = coefficients[i] * power(x, exponents[i]) + ans;
	}
	return ans;
}
```

Space = 8
Time = 2e+ne+5;

Challenge-D:

```
void displayPrimes(int n)
{
	Queue<int>numbers;
	Queue<int>primes;
	for(int i=2;i!=n+1;i++)
	{
		numbers.enQueue(i);
	}

	double limit = sqrt(n);
	int p ;
	do
	{
		p = numbers.deQueue();
		primes.enQueue(p);
		Queue<int>newQueue;
		while (!numbers.isEmpty())
		{
			if (numbers.getElementAtFront() % p==0)
```

```
                {
                        numbers.deQueue();
                }
                else
                {
                        newQueue.enQueue(numbers.deQueue());
                }
        }
        numbers=newQueue;
    } while (p < limit);

    while (!numbers.isEmpty())
    {
        primes.enQueue(numbers.deQueue());
    }
    while (!primes.isEmpty())
    {
        cout << primes.deQueue() << "\t";
    }
}
```

Using STL:

```
void displayPrimes(int n)
{
    queue<int> numbers;
    queue<int>primes;
    for(int i=2;i!=n+1;i++)
    {
        numbers.push(i);
    }

    double limit = sqrt(n);
    int p ;
    do
    {
        p = numbers.front();
        numbers.pop();
        primes.push(p);
        queue<int>newQueue;
        while (!numbers.empty())
        {
            if (numbers.front() % p==0)
            {
                numbers.pop();
            }
            else
            {
                newQueue.push(numbers.front());
                numbers.pop();
            }

        }
        numbers=newQueue;
    } while (p < limit);

    while (!numbers.empty())
    {
        primes.push(numbers.front());
        numbers.pop();
    }
    while (!primes.empty())
    {
        cout << primes.front() << "\t";
        primes.pop();
```

```
        }
}
```