CMP-210 Data Structures and Algorithms
BS Fall 2018
Quiz 2 – Online KEY

Roll-No: _ _ _ _ _ _ _ _ _ _ _ _

Issue Date: 4-Apr-2020
Marks: 27
Submission Date/Time: 7-Apr-2020 before 20:00 hours

Objective:

- This quiz will check your status of learning so far in the course by asking questions from easy to difficult based on knowledge learned so far.

Challenge-A: (4,4,2)

Give worst **time** and **space** bound of the following code function?

Part-A

$Time: O(N^3)$
$Space: O(N^2)$

Part-B

1)

$As\ per\ our\ convention\ used\ in\ class: \quad 3 = D_1, R = D_2, C = D_3$

$ByteNumber = i + j * 3 * C + k * 3 + 1$

$Row\ Start = i + j * 3 * C$
$Column\ Start = k * 3$
$+1\ to\ ajust\ the\ offset\ as\ the\ buyte\ number\ start\ from\ 1$

2)

$Lets\ say\ that\ given\ ByteNumber = BN$
$BN = BN - 1$
$i = BN\%3$
$j = BN/(3 * c)$
$k = BN/3\%C$

Part-C

CASE-A:	/(/	**ANSWER:** **TRUE** FALSE
CASE-B:	*/	**ANSWER:** TRUE **FALSE**
CASE-C:	/(+	**ANSWER:** **TRUE** FALSE
CASE-D:	/+ (+	**ANSWER:** TRUE **FALSE**

Challenge-B: (7)

```
double evaluatePolynomial(double coefficients[], int exponents[], int N, int x)
{
    double result =0;                                   1           8
    int expo = pow(2,exponents[0]);                     D           4
    int gap=0;                                          1           4
    result = result + coefficients[0]*expo;             1
    for (int i=1; i<N; i++)                             N           4
    {
        gap = exponents[i-1]-exponents[i];              N-1
        for (int k=1; k<=gap; k++)                      D+1         4
            expo=expo/2;                                D
        result = result + coefficients[i]*expo;         N-1
    }
    return result;                                      1
}                                                       Time        Space
```

D represents the degree of polynomial. Number of terms in a polynomial can be at max D+1
So I may take it as $O(D)$ time and $O(1)$ space.

Challenge-C: (5)

```
void indentCode(string inputFile)
{
    ofstream ofs("indented_"+inputFile);
    ifstream ifs(inputFile);
    int branchLevel=0;
    char ch;
    char prevCh='a';
    while(ifs.get(ch))
    {
        if (ch!='{' && ch!='}')
            ofs<<ch;
        else if (ch=='{')
        {
            if (!(prevCh=='\r' || prevCh=='\n'))
                ofs<<'\n';
            for (int i=1; i<=branchLevel; i++)
            {
                ofs<<'\t';
            }
            ofs<<'{';
            branchLevel++;
        }
        else if (ch=='}')
        {
            ofs<<'\n';
            for (int i=1; i<branchLevel; i++)
            {
                ofs<<'\t';
            }
            ofs<<'}';
            branchLevel--;
        }
        prevCh=ch;
    }
    ifs.close();
    ofs.close();
}
```

Challenge-D: (5)

Version-1: using user/programmer define structures

```
void displayPrimes(int N)
{
    if (N < 2)
        return;
    Queue<int> numbersQueue;
    for(int i=2; i<=N; i++)
        numbersQueue.enQueue(i);
    Queue<int> primeQueue;
    int limit = sqrt(N);
    int prime;
    do
    {
        prime = numbersQueue.deQueue();
        primeQueue.enQueue(prime);
        int noe = numbersQueue.getNoOfElements();
        for (int i=1; i<=noe; i++)
        {
            int x = numbersQueue.deQueue();
            if(x%prime!=0)
                numbersQueue.enQueue(x);
        }
    }
    while(prime<limit);

    cout<<"Total Primes: "<<primeQueue.getNoOfElements()+numbersQueue.getNoOfElements()<<endl;
    while(!primeQueue.isEmpty())
        cout<<primeQueue.deQueue()<<" ";
    while(!numbersQueue.isEmpty())
        cout<<numbersQueue.deQueue()<<" ";
}
```

Version-2: using STL provided structures

```
#include <queue>
void displayPrimes(int N)
{
    if (N < 2)
        return;
    queue<int> numbers;
    for (int i = 2; i <= N; i++)
        numbers.push(i);
    queue<int> primes;
    int limit = sqrt(N);
    int p;
    do
    {
        p = numbers.front();
        primes.push(p);
        int size = numbers.size();
        for (int i = 0; i < size; i++)
        {
            int num = numbers.front();
            if (num % p == 0)
            {
                numbers.pop();
            }
            else
            {
                numbers.pop();
                numbers.push(num);
            }
        }
    }
    while (p < limit);

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