Homework #01

01 April 2021 20 minutes

Name: ____ **Roll No**: ___________________________

Honor Code: By signing below, I am declaring under oath that
I will solve the quiz by myself solely and will not
Take any help from neighboring fellows. Signature

1. Compute the running time, with proper working, of the following code/functions as a function of n.

```
void fup(int n)
{
      int j = 1;
      int e = pow(2,n);
      while (j <= e)
      {
            cout << j << endl;
            j = j + 1;
      }
}

void fup2(int n)
{
      int j = 1;
      int e = pow(2,n);
      while (j <= e)
      {
            cout << j << endl;
            j = j * 2;
      }
}

void fbel1(int n)
{
      int j = 1;
      while (j <= n)
      {
            cout << "-------------\n";
            fup(n);
            j = j * 2;
      }
}
```

```
void fbel2(int n)
{
      int j = 1;
      while (j <= n)
      {
            cout << "--------------\n";
            fup(j);
            j = j * 2;
      }
}

void flst1(int n)
{
      int j = 1;
      while (j <= n)
      {
            cout << "--------------\n";
            fup(j);
            j = j + 2;
      }
}

void flst2(int n)
{
      int j = 1;
      while (j <= n)
      {
            int k = 1;
            while (j <= n)
            {
                  int p = n;
                  while (p >= 1)
                  {
                        cout << j+k+p << endl;
                        p = p / 2;
                  }
                  k = k + 1;
            }
            j = j + 2;
      }
}
```

---

2. Solve the following recurrences as a function of n. All have base case as T(1) = 1

- **$T(n) = T(n-1) + 1$**
- **$T(n) = T(n/2) + 1$**
- **$T(n) = T(n/2) + n$**
- **$T(n) = 2T(n/2) + 1$**
- **$T(n) = 2T(n/2) + n$**
- **$T(n) = 2T(n/2) + n^2$**

3. Formulate the recurrences for the following functions.

```
int fibonacci(int n)
{
      if (n==1 || n==2)
      {
            return 1;
      }
      int lsfib = fibonacci(n-1);
      int slfib = fibonacci(n-2);
      int cfib = slfib + lsfib;
      return cfib;
}
```

```
double power (double num, int p)
{
      if (p==1)
      {
            return num;
      }
      double t1 = power(num, p/2);
      double t2 = power(num, p-p/2);
      return t1 * t2;
}
```

```
int bsearch(double nums[], int li, int hi, double val)
{
      if (li > hi)
            return -1; // better to throw exception
      mi = (li + hi ) / 2;
      if (nums[mi] == val)
            return mi;
      else if (val < nums[mi])
            return bsearch(nums, li, mi, val);
      else if (val < nums[mi])
            return bsearch(nums, mi, hi, val);
}
```

--- The End ---