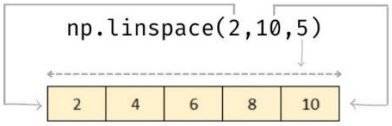
American International University – Bangladesh

CSC 2211: Algorithms Lab Exam

1. Write a warmup code to select your problem set.

```
#include<bits/stdc++.h>

using namespace std;

int main(){

    int middleId = 1997;

    cout<<(middleId%3+1)<<endl;

return 0;

}
```

2. According to your output, select the same number of problems to solve in the lab exam.
   i. Maximum Pair Wise Product and Maximum Pair Wise Product Fast
   ii. Selection Sort and Merge Sort
   iii. Insertion sort and Quick Sort
3. Generate 100000 and 1000000 random data for the test cases.
4. Write a function generates linearly spaced vectors.



Maximum Pair Wise Product

Algorithm 4 Maximum Pairwise Product

```
1: procedure MaxPairwiseProductNaive(A, n)
2:     product← 0
3:     for i ← 1, n do
4:         for j ← 1, n do
5:             if i! = j then
6:                 if product < A[i] ∗ A[j] then
7:                     product = A[i] * A[j]
8:                 end if
9:             end if
10:        end for
11:    end for
12:    return product
13: end procedure
```

Maximum Pair Wise Product Fast

Algorithm 5 Maximum Pairwise Product Fast

```
1:  procedure MaxPairwiseProductFast(A, n)
2:      index_1 ← 0
3:      for i ← 0, n − 1 do
4:          if A[i] ≥ A[index_1] then
5:              index_1 = i
6:          end if
7:      end for
8:      index_2 ← 0
9:      for i ← 1, n − 1 do
10:         if i! = index_1 && A[i] ≥ A[index_2] then
11:             index_2 = i
12:         end if
13:     end for
14:     return A[index_1] ∗ A[index_2]
15: end procedure
```

Selection Sort

Algorithm 5 Selection Sort

```
1:  procedure SelectionSort(A, n)
2:      for i ← 0, n − 1 do
3:          iMin ← i
4:          for j ← i + 1, n − 1 do
5:              if A[j] < A[iMin] then
6:                  iMin = j
7:              end if
8:              swap(A[iMin], A[i])
9:          end for
10:     end for
11: end procedure
```

Merge Sort

Algorithm 8 Merge

```
1:  procedure MERGE(A, left, mid, right)
2:      n1 = mid − left + 1
3:      n2 = right − mid
          L[1....n1]andR[1....n2]
4:      for i ← 0, n1 − 1 do
          L[i] ← A[left + i]
5:      end for
6:      for j ← 0, n2 − 1 do
          R[j] ← A[mid + 1 + j]
7:      end for
8:      i ← 0, j ← 0, k ← left
9:      while i ≤ n1 − 1&j ≤ n2 − 1 do
10:         if L[i] < R[j] then
11:             A[k + +] ← L[i + +]
12:         else
13:             A[k + +] ← R[i + +]
14:         end if
15:     end while
16:     while i <= n1 − 1 do
17:         A[k + +] ← L[i + +]
18:     end while
19:     while j <= n2 − 1 do
20:         A[k + +] ← R[j + +]
21:     end while
22: end procedure
```

Algorithm 9 Merge Sort

```
1: procedure MERGESORT(A, left, right)
2:     if left < right then
3:         mid = (left + right)/2
4:         MERGESORT(A, left, mid)
5:         MERGESORT(A, mid + 1, right)
6:         MERGE(A, left, mid, right)
7:     end if
8: end procedure
```

Bubble sort

Algorithm 4 Bubble Sort

```
1: procedure BubbleSort(A, n)
2:     for k ← 0, n − 1 do
3:         for i ← 0, n − 1 do
4:             if A[i] > A[i + 1] then
5:                 swap(A[i], A[i + 1]
6:             end if
7:         end for
8:     end for
9: end procedure
```

Quick Sort

Algorithm 10 partition

```
1: procedure PARTITION(A, start, end)
2:     pivot = A[end]
3:     pIndex = start
4:     for i ← start, end − 1 do
5:         if A[i] < pivot then
6:             swap(A[i], A[pIndex]) pIndex + +
7:         end if
8:     end for
9:     swap(A[pIndex], A[end]) return pIndex
10: end procedure
```

Algorithm 11 Quick Sort

```
1: procedure QuickSort(A, start, end)
2:     if start >= end then
3:         pIndex =PARTITION(A, start, end)
4:         QuickSort(A, start, pIndex − 1)
5:         QuickSort(A, pIndex + 1, end)
6:     end if
7: end procedure
```