Data Structure Part 1

1.1 Objectives

i. To be able to link knowledge of data structures in programming.

1.2 Requirements

i. Laptop/ notebook /smart phone
ii. Internet
iii. Pen/Pencil
iv. Paper

1.3 Tasks

1. Create a group with at most 2 members in Ifolio --> Groups --> Groups for Tutorial 1.
2. In groups, discuss what are the output of the Program 1: Array and Vector

```
#include <cstdio>
#include <vector>
using namespace std;

int main() {
  int arr[5] = {7,7,7};   // initial size (5) and initial value
{7,7,7,0,0}
  vector<int> v(5, 5);    // initial size (5) and initial value
{5,5,5,5,5}

  printf("arr[2] = %d and v[2] = %d\n", arr[2], v[2]);

  for (int i = 0; i < 5; i++) {
    arr[i] = i;
    v[i] = i;
  }

  printf("arr[2] = %d and v[2] = %d\n", arr[2], v[2]);

  // arr[5] = 5;      // static array will generate index out of bound
error
  // uncomment the line above to see the error

  v.push_back(5);                         // but vector will resize
itself
  printf("v[5] = %d\n", v[5]);
// 5

  return 0;
```

Program 1: Array and Vector

3. In groups, discuss what is deque and what are the output of the Program 2: Stack and Queue

```
#include <cstdio>
#include <stack>
#include <queue>
using namespace std;

int main() {
  stack<char> s;
  queue<char> q;
  deque<char> d;

  printf("%d\n", s.empty());                  // currently s is empty, true (1)
  printf("==================\n");
  s.push('a');
  s.push('b');
  s.push('c');
  // stack is LIFO, thus the content of s is currently like this:
  // c <- top
  // b
  // a
  printf("%c\n", s.top());
  s.pop();                                    // pop topmost
  printf("%c\n", s.top());                              //
  printf("%d\n", s.empty());        // currently s is not empty, false (0)
  printf("==================\n");

  printf("%d\n", q.empty());             // currently q is empty, true (1)
  printf("==================\n");
  while (!s.empty()) {                 // stack s still has 2 more items
    q.push(s.top());                    // enqueue 'b', and then 'a'
    s.pop();
  }
  q.push('z');                                // add one more item
  printf("%c\n", q.front());                              //
  printf("%c\n", q.back());                               //

  // output 'b', 'a', then 'z' (until queue is empty), according to the insertion order above
  printf("==================\n");
  while (!q.empty()) {
    printf("%c\n", q.front());                  // take the front first
    q.pop();                          // before popping (dequeue-ing) it
  }

  printf("==================\n");
  d.push_back('a');
  d.push_back('b');
  d.push_back('c');
  printf("%c - %c\n", d.front(), d.back());              //  d.push_front('d');
  printf("%c - %c\n", d.front(), d.back());              //
  d.pop_back();
  printf("%c - %c\n", d.front(), d.back());            //
  d.pop_front();
  printf("%c - %c\n", d.front(), d.back());            //

  return 0;
}
```

Program 2: Stack and queue

1.4 Evaluation

Items	
Attendance	**10**
Answer	**30**
Presentation	**10**
Total	**50**