

**Data Structures & Algorithms (DSA)**

Year 2/3 (2020/21), Semester 4/6

## SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in Cybersecurity & Digital Forensics

Diploma in Information Technology

**TEST 1 – SOLUTION DOCUMENT**

INSTRUCTIONS TO CANDIDATES:

1. Write your Student Number, Name and Module Group CLEARLY in the boxes provided below.
2. Provide your answers to the questions in the Test 1 paper in this document.
3. Save this file as "Test1 – s1234567 Solution.docx" where s1234567 is your student number.
4. Map to network drive: [**\\ictspace.ict.np.edu.sg\DSATest1\**](file:///\\ictspace.ict.np.edu.sg\DSATest1\)
5. Copy this solution file into the network drive.

**ictspace.ict.np.edu.sg > DSATest1 > group > studentID**

|  |  |
| --- | --- |
| **Student Number:** | **Seat Number:** |
| **Student Name:** | **Module Group:** |

**GRADE**

There are 3 questions. Answer ALL questions (100 marks).

Write your solutions to the questions in the space allocated for each question.

Question 1 – Solution (50 marks)



|  |  |
| --- | --- |
| (a) | void List::rotate(int num)  { // create a tempList to point to the first num Nodes  if (num <= 0)  return;  Node\* tempList = firstNode;  Node\* current = firstNode;  for (int i = 0; i < num-1; i++)  current = current->next;  // let firstNode point to the (num+1) Node (make adjustments)  firstNode = current->next;  current->next = NULL; // terminate tempList  // traverse to the end of the existing list  current = firstNode;  while (current->next != NULL)  current = current->next;  current->next = tempList; // add tempList to the end  } |
|  | (20 marks) |
| (b) | void List::rotate(int num)  {  for (int j = 0; j < num; j++) // set up loop  {  ItemType temp = items[0]; // set up for looping  int i;  for (i = 0; i < size - 1; i++) // set up nested loop  items[i] = items[i + 1]; // shifting  items[i] = temp; // copy front item to back  }  } |
|  | (20 marks) |
| (c) | Time complexity:  Pointer-based: O(n)  Just need to traverse down the list to make adjustments.  Array-based: O(n2)  Need a nested loop to shift each item n times for each rotation. |
|  | (10 marks) |

Question 2 – Solution (25 marks)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (a) | front back   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 10 |  |  | 4 |  |   front back   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 10 |  |  | 6 |  |  | 4 |  |   front back   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 10 |  |  | 6 |  |  | 4 |  |  | 3 |  |   front back   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 10 |  |  | 6 |  |  | 5 |  |  | 4 |  |  | 3 |  | |
|  | (10 marks) |
| (b) | bool PQueue::enqueue(ItemType item, PriorityType p)  {  // create a new node to store data  Node\* newNode = new Node;  newNode->item = item;  newNode->priority = p;  newNode->next = NULL;  // insert the new node  if (isEmpty()) // enqueue at the front  frontNode = newNode;  else if (frontNode->priority < p) // enqueue before first node  {  newNode->next = frontNode;  frontNode = newNode;  }  else  { // traverse to the correct place to insert  Node\* temp = frontNode;  while (temp->next != NULL && temp->next->priority > p )  temp = temp->next;  // insert  newNode->next = temp->next;  temp->next = newNode;  }  return true;  } |
|  | (15 marks) |

Question 3 – Solution (25 marks)



|  |  |
| --- | --- |
| (a) | int count\_digit(int number)  {  if (number < 10) // base case  return 1;  number = number / 10;  return 1 + count\_digit(number); // recursive step  } |
|  | (10 marks) |
| (b) | bool isPalindrome(int num)  {  int num\_digits = count\_digit(num);  if (num\_digits == 1) // base case  return true;  int power = 10; // get left-most digit  for (int i = 1; i < num\_digits-1; i++)  power \*= 10;  int left\_digit = num / power;  int right\_digit = num % 10; // get right-most digit  if (left\_digit == right\_digit) // recursive step  {  num = (num % power) / 10;  return isPalindrome(num);  }  else  return false;  } |
|  | (15 marks) |

**– End of Document –**