

**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: s10198161** | **Seat Number:** |
| **Student Name: Tan Jia Shun** | **Module Group: P02** |

**GRADE**

**B+**

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)  {  if (size != 0 && num > 0) {  Node\* item;  for (int i = 0; i < num; i++) {  item = firstNode;  Node\* lastnode = NULL;  Node\* nextnode = NULL;  Node\* tmp = firstNode;  for (int k = 0; k < size; k++) {  if (k == size - 1) {  lastnode = tmp;  }  else {  if (k == 1) {  nextnode = tmp;  }  tmp = tmp->next;  } tmp moves to the end of the list  }  lastnode->next = item; lastnode points to firstNode  lastnode = lastnode->next; lastnode moves to next node  which is the second node.  lastnode->next = NULL; this breaks lastnode from the rest  So the list is separated, broken into 2, with one part firstNode and second node, and the other part is the rest.  firstNode = nextnode; of list.  } firstNode now points to nextnode which is the firstNode.  }  } |
|  | (20 marks) |
| (b) | void List::rotate(int num)  {  if (size != 0 && num > 0) {  for (int i = 0; i < num; i++) {  items[size-1] = items[0];  for (int k = 0; k < size-1; k++) {  items[k] = items[k + 1];  }  items[size] = NULL;  }  }  }🗸 |
|  | (20 marks) |
| (c) | Pointer based will take longer to run as it need to loop the whole firstnode to look for the second node and the last node, than it can start rotating but for array base, we can just set the value at the first list to the last list using items[size] which will the last list slot. Than we just need to loop to move the list Infront by 1 slot.🗴  They have the same time complexity. |
|  | (10 marks) |

Question 2 – Solution (25 marks)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (a) | |  |  | | --- | --- | | Front | Back | |  |  |      |  |  | | --- | --- | | 10 |  |  |  |  | | --- | --- | | Front | Back | |  |  |      |  |  |  |  |  | | --- | --- | --- | --- | --- | | 10 |  |  | 4 |  |  |  |  | | --- | --- | | Front | Back | |  |  |      |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 10 |  |  | 4 |  |  | 6 |  |  |  |  | | --- | --- | | Front | Back | |  |  |      |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 10 |  |  | 4 |  |  | 6 |  |  | 3 |  |  |  |  | | --- | --- | | Front | Back | |  |  |      |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 10 |  |  | 4 |  |  | 6 |  |  | 3 |  |  | 5 |  |   Nodes not added in order of priority. |
|  | (10 marks) |
| (b) | bool PQueue::enqueue(ItemType item, PriorityType p)  {  Node\* tmp = new Node;  tmp->item = item;  tmp->priority = p;  tmp->next = NULL;🗸  if (frontNode == NULL) {  frontNode = tmp;  }🗸  else {  Node\* loopNode = frontNode;  while (true)  {  if (loopNode->next->priority < p) {  tmp->next = loopNode->next;  loopNode->next = tmp;  return true;  }  else if (loopNode->next == NULL) { this should be before  loopNode->next = tmp; the other if  return true; priority may be bigger  }  else {  loopNode = loopNode->next;🗸  }  }  }  return true;  } |
|  | (15 marks) |

Question 3 – Solution (25 marks)

|  |  |
| --- | --- |
| (a) | int counter(int number, int count) {  int a = number / pow(10,count);  if (a != 0) {  return counter(number, count+1);  }  else {  return count;  }  }🗸  int count\_digit(int number) {  return counter(number, 1);  }🗸 |
|  | (10 marks) |
| (b) | bool checkPalindrome(string newNum, int length, int i) {  if (length == i) {  return true;  }  else if (newNum[i] != newNum[length - 1 - i]) {  return false;  }  else {  return checkPalindrome(newNum, length, i+1);  }  }  bool isPalindrome(int num) {  int length = count\_digit(num);  string newNum = to\_string(num);  return checkPalindrome(newNum, length, 0);  You should not change num to string! But this works! -2 mark  } |
|  | (15 marks) |

**– End of Document –**