THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE

EXAMINATIONS OF APRIL/MAY 2017

Code and Name of Course: COMP1603 — Computer Programming III

Paper:

Date and Time: Tuesday 16th May 2017 1 pm

Duration: 2 Hours

INSTRUCTIONS TO CANDIDATES: This paper has 5 pages and 3 questions

Answer all questions

1. (a) Consider the program shown below. Give the output of the program.

```
#include <iostream>
using namespace std;
int main() {
    void test (int *ptr, int n);
    int num[5];

    for (int j = 1; j < 5; j++)
       num[j] = 5 * j;

    test(num, 5);

    for (int j = 0; j < 5; j++)
      cout << "num[" << j <<"] is " << num[j] << endl;

    return 0;
}

void test(int *ptr, int max) {
   for (int j = 0; j < max-1; j++) {
      (*ptr) += 2;
       ptr++;
   }
   *(ptr-1) = 4;
 }
```

[6]

(b) Assume that an integer array **nums** has been declared and filled to capacity with 100 integers.

Write code to read the values from **nums** above and place them in a Stack, **s**. Next, pop the stack values one at a time and find the sum of every other value that is popped i.e. the 1st value is popped and added to the sum, then the 2nd value is popped and discarded, next, the 3rd value is popped and added to the sum and then the 4th value is popped and discarded and so on until the end of the stack. Print the final sum.

[Question 1 continues on the following page]

You may assume the existence of the usual Stack functions. Some prototypes are listed below.

```
Stack * initStack();
bool isEmpty (Stack * s);
bool isFull (Stack * s );
int peek (Stack * s);
void push (Stack * s, int n);
int pop (Stack * s);
```

[6]

(c) (i) Give the declarations for a linked queue of integers. [2]

(ii) Write the code for the **enqueue** and **dequeue** functions. You may assume the existence of the following functions:

```
Queue * initQueue();
bool isEmpty(Queue * q);
int peek(Queue * q);
Node * createNode (int n);
```

[6]

Total Marks for Question 1 is 20

2. (a) What output is produced by the call **mystery (10)** of the following recursive function? [6]

```
void mystery (int n) {
   if (n > 5) {
    mystery (n - 2);
    cout << n << "\t" << endl;
    mystery (n - 3);
    cout << n << "\t" << endl;
   }
}
```

[Question 2 continues on the following page]

For parts (b) and (c) below,
assume that **top** points to a linked list of integers.

NB: The linked list may be empty.

The declarations for the nodes of the linked list follow:

```
struct Node {
   int data;
   Node * next;
};
```

(b) Write a recursive function **count5** to count the number of times the integer **5** occurs in a linked list. The function prototype is

int count5 (Node *top); [4]

(c) Write a recursive function that accepts two parameters: an integer **n** and a pointer **top**. The function adds **n** to all the elements of the linked list. E.g. If **n** is 5 and the first node contains 6, then the data in the first node is changed to 6 + 5 = 11 etc. The function prototype is:

void recAddn (Node *top, int n); [5]

(d) Write a recursive function to read a line of data terminated by $, character by character, and print it with the characters reversed.

E.g. If the input is abcd$, the function prints dcba

Note: No array or linked list storage must be used. [5]

Total Marks for Question 2 is 20

3. This question is based on linked lists of integers.
The declarations for the nodes of the linked list follow:

```
struct Node {
   int data;
   Node * next;
};
```

(a) Write a function that accepts as input two positive integers **m** and **n** (where **m** <= **n**) and returns the sum as well as the product of the integers from **m** (inclusive) to **n** (inclusive). Store the sum in a variable called **sum1** and the product in a variable called **product1**.

Note: Your code must only use pointer variables. This means that you must create all the memory locations required by the code. Exceptions would be that the inputs to the function (**m** and **n**) are allowed to be integers. [4]

(b) Write a function, `insertAtTail` that accepts a pointer to the top of a linked list and an integer **n**. The function inserts **n** at the end of the linked list. (Do not assume that a **getLast** function exists).

The prototype of the function is

```
Node *insertAtTail (Node * top, int n);
```

[6]

(c) Write a function `removeDuplicates`, which takes the top of an **unsorted** linked list as input and deletes any duplicate nodes from the list. Return the new list after the duplicates have been removed.

The function prototype is

```
Node *removeDuplicates(Node *top);
```

[10]

Total Marks for Question 3 is 20

End of Question Paper (Total Marks 60)