THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE

EXAMINATIONS OF APRIL/MAY 2018

Code and Name of Course: COMP1603 — Computer Programming III

Paper:

Date and Time:

Duration: 2 Hours

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

Answer all questions
All Questions are not Equally Weighted

1. (a) Consider the program shown below. Assume that it compiles successfully. Give the output of the program.

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main() {

    int x, y;
    int *px, *py, *z;
    px = &x;
    py = &y;
    x = 3;
    y = 7;
    z = &x;
    *z = *z - y;

    cout <<  " 1. *z is " << *z << endl;
    *px = *px + 10;
    cout <<  " 2. The value of *px is " <<  *px << endl;
    x = x + 100;
    cout <<  " 3. The value of *py is " <<  *py << endl;
    cout <<  " 4. The value of x is " << x << endl;

    *py = *py + 5;
     y = *py + y;
    *px = x + y;
    cout <<  " 5. The value of x is " << x << endl;
    cout <<  " 6. The value of y is " << y << endl;

    int arr[] = {2,4,6,8};
    int *ptr = arr;
    for (int j=0; j < 3; j++) {
      (*ptr) *= 2;
      ptr++;
    }
    for (int j=0; j < 4; j++)
      cout << arr[j] << '\t';

}
```

[10]

[Question 1 continues on the next page]

(b) What is an Abstract Data Type (ADT)? [2]

(c) Assume that an integer array **nums** has been declared and filled to capacity with 200 integers.

Write code to obtain the values from **nums** above and place them in a Queue, **q**. Next, dequeue the values from **q** one at a time and place the values in a stack, **s**. Finally, pop the stack **s** and find the product of the values popped. Display the final product.

You may assume the existence of the usual Stack and Queue 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);

Queue * initQueue ();
bool isEmpty (Queue * q);
int peek (Queue * q);
void enqueue (Queue * q, int n);
int dequeue (Queue * q);
```

[8]

Total marks 20

2. (a) What output is produced by the call **mystery (11)** of the following recursive function? [6]

```
void mystery (int n) {
   cout << n << "\t" << endl;
   if (n < 0)
      return;
   else if (n > 4)
     mystery (n - 2);
   else mystery (n - 3);
}
```

[Question 2 continues on the next page]

(b) Write a recursive function **sum** to find and return the sum of the integers in a linked list. The function prototype is:

int sum (Node *top);

Node declaration:

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

[4]

(c) This part is based on a linked list of integers.
The declarations for the nodes of the linked list follow:

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

Write a function **insertInOrder** which accepts a pointer to a linked list and an integer **key**. The function inserts **key** into the linked list so that the linked list is maintained in **ascending** order. Return a pointer to the updated list. The prototype for the function is:

Node *insertInOrder (Node * top, int key)

Note that the linked list may be empty initially. [5]

Total marks 15

[Please turn the page]

3. (a) What is the postfix form of the following expression? [2]
6 *8 / (4 * 3) + 10 - (8/4/2)

(b) Write code to evaluate a postfix expression which is stored in **expr_arr**, an array of **char**. Only single digit operands can appear in **expr_arr**. The postfix form that is stored in **expr_arr** is terminated by the character '$'. The operators '-', '+', '*' are possible, but not '/'. Assume that the postfix expression is valid.

Example:

expr_arr	5	2	+	6	3	-	*	$

The above expression evaluates to 21.

You may assume the existence of the main stack functions.
Sample stack function prototypes:

```
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) Write a function **intersect** which finds the common elements of two linked lists **list1** and **list2** and stores the common elements in **list3**. Return the newly created list. Assume that **list1** does not have any duplicates. **list2** may have duplicates. **list3** must not have any duplicates. The function prototype is:

Node *intersect (Node *list1, Node *list2)

The declarations for the nodes of the linked list follow:

```
struct Node {
  int data;
  Node * next;
};
```
[12]

Total Marks 20

End of Question Paper