CE1007/CZ1007 – DATA STRUCTURES

Assignment Questions: Stacks and Queues

Information:

Program templates for questions 1-4 are given as separated files (**Q1_template_SQ.c, Q2_template_SQ.c**, **Q3_template_SQ.c**, and **Q4_template_SQ.c**). You **must use** them to implement your functions. The program contains a *main()* function, which includes a switch statement to execute different functions that you should implement. Each function can be called multiple times depending on the user's choice. You need to submit your code to the Automated Programming Submission and Assessment System (APAS) for testing. Deadline for program submission: **November 4th, 2019 (Monday) 11.59 pm**.

Assignment Questions (1-4)

1. Write a C function `createQueueFromLinkedList()` to create a queue (linked-list-based) by enqueuing all integers which are stored in the linked list. The first node of the linked list is enqueued first, and then the second node, and so on. Remember to empty the queue at the beginning, if the queue is not empty.

   After the queue is built, write another C function `removeOddValues()` to remove all odd integers from the queue. Note that you should only use `enqueue()` or `dequeue()` when you add or remove integers from queues.

   The function prototypes are given as follows:

```
void createQueueFromLinkedList(LinkedList *ll , Queue *q);
void removeOddValues(Queue *q)
```

   A sample input and output session is given below (if the current linked list is 1 2 3 4 5):

```
The resulting linked list is:  1 2 3 4 5

Please input your choice(1/2/3/0): 2
The resulting queue is: 1 2 3 4 5

Please input your choice(1/2/3/0): 3
The resulting queue after removing odd integers is: 2 4
```

2. Write a C function `reverseStack`() that reverses a stack using **a queue**. Note that the `reverseStack`() function only uses `push`()and `pop`()when adding or removing integers from the stack, and only uses `enqueue`() and `dequeue`() when adding or removing integers from the queue.

   The function prototypes are given as follows:

```
void reverseStack(Stack *s);
```

   For example, if the stack is (**1, 2, 3, 4, 5**) the resulting stack will be (**5, 4, 3, 2, 1**).

   A sample input and output session is given below:

```
1: Insert an integer into the stack;
2: Reverse the stack;
0: Quit;
Please input your choice(1/2/0): 1
Input an integer that you want to insert into the stack: 8
The resulting stack is: 8
Please input your choice(1/2/0): 1
Input an integer that you want to insert into the stack: 5
The resulting stack is: 5 8
Please input your choice(1/2/0): 1
Input an integer that you want to insert into the stack: 7
The resulting stack is: 7 5 8
Please input your choice(1/2/0): 1
```

```
Input an integer that you want to insert into the stack: 9
The resulting stack is: 9 7 5 8
Please input your choice(1/2/0): 2
The resulting stack after reversing its elements is: 8 5 7 9
Please input your choice(1/2/0): 0
```

3. Write a C function write a function `isStackPairwiseConsecutive()` that checks whether numbers in the stack are pairwise consecutive or not. Note that the `isStackPairwiseConsecutive()` function **only** uses `push()` and `pop()` when adding or removing integers from the stack.

The function prototype is given as follows:

```
int isStackPairwiseConsecutive(Stack *s);
```

Sample test cases are given below:

Test case 1
The stack is: 16 15 11 10 5 4
The stack is pairwise consecutive

Test case 2
The stack is: 16 15 11 10 5 1
The stack is not pairwise consecutive

Test case 3
The stack is: 16 15 11 10 5
The stack is not pairwise consecutive

A sample input and output session is given below (if the current stack is 16 15 11 10 5 4):

```
1: Insert an integer into the stack:
2: Check the stack is pairwise consecutive:
0: Quit:

Please input your choice(1/2/0): 1
Input an integer that you want insert into the stack: 4
The stack is: 4
Please input your choice(1/2/0): 1
Input an integer that you want insert into the stack: 5
The stack is: 5 4
Please input your choice(1/2/0): 1
Input an integer that you want insert into the stack: 10
The stack is: 10 5 4
Please input your choice(1/2/0): 1
Input an integer that you want insert into the stack: 11
The stack is: 11 10 5 4
Please input your choice(1/2/0): 1
Input an integer that you want insert into the stack: 15
The stack is: 15 11 10 5 4
Please input your choice(1/2/0): 1
Input an integer that you want insert into the stack: 16
The stack is: 16 15 11 10 5 4

Please input your choice(1/2/0): 2
The stack is pairwise consecutive.
```

4. Write a C function `popToQueue()` that pop (insert) the stack integers into the queue in front of a matching value. Note that the `popToQueue()` function only uses `pop()` when removing integers from the stack and only uses `enqueue()` and `dequeue()` when adding or removing integers into/from the queue. A new queue is returned.

The function prototype is given as follows:

```
Queue popToQueue(Queue *q, Stack *s, int value);
```

A sample input and output session is given below (if the current queue is 1 2 3 4):

```
1: Insert an integer into the queue and stack;
```

```
2: Pop the stack to the queue;
0: Quit;
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 7
The queue is: 7
The stack is: 7
Please input your choice(1/2/0): 1

Input an integer that you want insert into the queue and stack: 9
The queue is: 7 9
The stack is: 9 7
Please input your choice(1/2/0): 2
The queue is 7 9
The stack is 9 7
Enter the integer in the queue that you want to pop the stack to : 9
The resulting queue after popping the stack elements is: 7 9 7 9
Please input your choice(1/2/0): 2
The queue is Empty
The stack is Empty
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 5
The queue is: 5
The stack is: 5
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 7
The queue is: 5 7
The stack is: 7 5
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 8
The queue is: 5 7 8
The stack is: 8 7 5
Please input your choice(1/2/0): 2
The queue is 5 7 8
The stack is 8 7 5
Enter the integer in the queue that you want to pop the stack to : 7
The resulting queue after popping the stack elements is: 5 8 7 5 7 8
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 9
The queue is: 9
The stack is: 9
Please input your choice(1/2/0): 2
The queue is 9
The stack is 9
Enter the integer in the queue that you want to pop the stack to : 9
The resulting queue after popping the stack elements is: 9 9
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 8
The queue is: 8
The stack is: 8
Please input your choice(1/2/0): 2
The queue is 8
The stack is 8
Enter the integer in the queue that you want to pop the stack to : 9
The resulting queue after popping the stack elements is: 8
Please input your choice(1/2/0):
```

If same digit in the queue:

```
1: Insert an integer into the queue and stack;
2: Pop the stack to the queue;
0: Quit;
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 3
The queue is: 3
The stack is: 3
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 2
The queue is: 3 2
The stack is: 2 3
Please input your choice(1/2/0): 1
```

```
Input an integer that you want insert into the queue and stack: 4
The queue is: 3 2 4
The stack is: 4 2 3
Please input your choice(1/2/0): 1
Input an integer that you want insert into the queue and stack: 2
The queue is: 3 2 4 2
The stack is: 2 4 2 3
Please input your choice(1/2/0): 2
The queue is 3 2 4 2
The stack is 2 4 2 3
Enter the integer in the queue that you want to pop the stack to : 2
The resulting queue after popping the stack elements is: 3 2 4 2 3 2 4 2
Please input your choice(1/2/0): 2
The queue is Empty
The stack is Empty
Please input your choice(1/2/0): 0
Press any key to continue . . .
```