Data structures and Algorithms LAB – BSDSF21
(Morning and Afternoon)

Lab 06 – 14-02-2023

The code related to the stack implemented through linked nodes as discussed in class is provided here for your reference. Using that, you have to complete the following task related to the stack. **10 each**

1. Create and test a function that check weather a sting passed to it as parameter is a palindrome or not. Palindrome is word that read same from both side, i.e., left to right or right to left. Examples of palindromes are abba, pop, level noon, abbabbababbabba, etc.
2. Create and test a function to match the balance of the parenthesis in an expression taken as parameter.
3. Create and test a function to evaluate a postfix expression for operation +, -, *, /.

Using two parallel stacks for x and y coordinates respectively, solve the following back tracking problem. Logic for solving a backtracking problem using a stack is as follows: push the starting known data item onto the stack, and while (use loop here) stack is not empty, pop a value from stack and save it in some variable, processing it and if necessary, push the connected/related data items onto the stack. **20 each**

4. Implement the following functions recursively and also write their tester main logic.

void replace(data, height, width, sr, sc, bc, fc)

Here, the function is similar to bucket filling function of paint software; ***data*** is a 2D array of size ***height*** times ***width*** (have number of rows equal to its height and number of columns equal to its width), ***bc*** is the value at ***sr*** and ***sc*** (the starting point, row and column number within array ***data***) before the replace function called from a main logic function. The function have to recursively replace all connected neighboring elements of array have the same value ***bc*** stored in them with the value ***fc***. The base case may be taken if the neighbors do not contain the same ***bc***, or boundary of the array is crossing.

Following is an illustration of replace(matrix, 8, 9, 3, 5, 2, 11)
Where matrix is declared as **array of 8 rows and 9 columns with data as on below left.**

1	1	1	1	1	**2**	1	1	2
2	2	8	**2**	**2**	**2**	**2**	1	1
2	8	8	**2**	**2**	5	7	8	2
2	8	8	**2**	9	**2**	**2**	8	3
4	4	0	**2**	9	**2**	6	**2**	**2**
0	4	**2**	**2**	9	**2**	**2**	**2**	5
9	4	**2**	**2**	**2**	**2**	**2**	**2**	4
0	4	4	4	4	4	5	4	4
Before replace								

1	1	1	1	1	**11**	1	1	2
2	2	8	**11**	**11**	**11**	**11**	1	1
2	8	8	**11**	**11**	5	7	8	2
2	8	8	**11**	9	**11**	**11**	8	3
4	4	0	**11**	9	**11**	6	**11**	**11**
0	4	**11**	**11**	9	**11**	**11**	**11**	5
9	4	**11**	**11**	**11**	**11**	**11**	**11**	4
0	4	4	4	4	4	5	4	4
After replace								

Data structures and Algorithms LAB – BSDSF21
(Morning and Afternoon)

Lab 06 – 14-02-2023

Similar to the provided stack class, design a Queue class as discussed in class session. **10 each**

5. Code the Queue class implemented through linked nodes.
6. Code for the testing the various functionalities of the Queue class.

Simulate a printer queue as follows for a printer with printed speed 5 pages per tick: **20**

7. In the main loop doing tick tick tick …, with too many iterations (initially take them between 10 and 100) generate a random number. If the number is divisible by 5 or 13 or 67, take its remainder (modulus) by dividing with 61 in a variable named pages and insert value of pages into the queue. Irrespective of the random number generated in the tick tick loop specified earlier:
   - If printer is free and queue is not empty, extract a task from the queue (pages to print). Assign task to the printer, which takes some tick to print it completely.
   - If printer is not free, consume the tick just as it prints pages and state of queue remain unchanged.
   - If printer is free and queue is empty, do nothing in the tick.

The output of the program should be similar to the following:

```
At tick 4, task 1 of 15 pages received for printing.
At tick 4, task 1 starts printing.
At tick 6, task 1 completed.
At tick 18, task 2 of 52 pages received for printing.
At tick 18, task 2 starts printing.
At tick 21, task 3 of 4 pages received for printing.
At tick 28, task 2 completed.
At tick 29, task 3 starts printing.
At tick 29, task 3 completed.
.
.
.
.
.
.
.
.
```

*********** The end ***********