Worksheet # 9 Data Structures – Stacks and Queues. CIS2500

Structure definition and some Stack functions required for Questions 1, 2 and 3:

```
8  typedef struct {
9
10     int stack[MAX];          // stack as an array
11     int top;                 // top position of the stack
12
13 } Stack;
14
15 /* removes the top element from stack s and returns it*/
16 int pop (Stack * s);
17
18 /* pushes element onto stack s*/
19 void push (int element, Stack * s);
20
21 /* returns the top-most element of stack s - does not remove it*/
22 int peek (Stack s);
23
24 /* prints the stack elements s */
25 void printStack(Stack s);
26
27 /* returns 1 if stack s is empty; 0 if not*/
28 int isEmpty (Stack s);
```

```
Q1:
push(11, &s);
e = push(&s); //popping 11

push(12, %s);
push(13, &s);
e = pop(&s); //poping 13

push(14, &s);
push(15, &s);
e = pop(&s); //poping 15
e = pop(&s); //poping 14
e = pop(&s); //poping 12
```

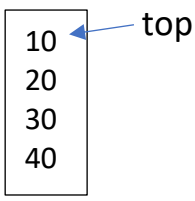
```
Q2:

push(pop(&s1), &s2); // pop 20 and put it into s2
push(pop(&s1), &s3); // pop 30 and put it into s3
push(pop(&s2), &s1); // pop 20 and put it into s1
push(pop(&s2), &s3); // pop 10 and put it into s3
push(pop(&s1), &s2); // pop 20 and put it into s2
push(pop(&s1), &s3); // pop 40 and put it into s3
push(pop(&s2), &s3); // pop 20 and put it into s3
```

Question 1: Suppose that you are required to push values 11, 12, 13, 14, 15 (in that order) to a stack s. But the values can be popped at any time. Write a sequence of pop and push operations such that the values are popped from the stack in the following order: 11, 13, 15, 14, 12. You may use the prototypes given on line 16 and line 19 to write your pop and push operations.

Question 2: Suppose there are 3 stacks s1, s2, s3 with a starting state as shown below. Write a sequence of push and pop operations using the given prototypes that take you from the start state to the end state. You are allowed to use only the given 3 stacks – you cannot use any other variable or data structure.

Start state of s1:



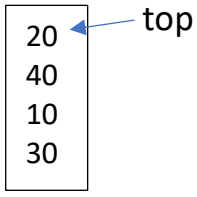
Start state of s2: Empty

Start state of s3: Empty

Finish state of s1: Empty

Finish state of s2: Empty

Finish state of s3:



Question 3: Trace the code and show the state of the stack after call to function *change* on 47.

```
33  int main () {
34
35      Stack s;            // declare a stack variable
36      int element;
37
38      s.top = -1;            // set the top of the stack
39
40      /* pushing 4 elements onto stack s*/
41      push (3, &s);
42      push (4, &s);
43      push (10, &s);
44      push (20, &s);
45
46
47       change (&s);
48       printStack (s);
49       return 0;
50
51  }
52
53  void change (Stack * s1) {
54
55      Stack s2;
56      int first, second;
57
58      s2.top = -1;
59
60      while (!isEmpty (*s1)) {
61
62         first = pop (s1);
63         if (isEmpty (*s1)) {
64
65            push (first, &s2);
66         }
67         else {
68
69            second = pop (s1);
70
71            push (second, &s2);
72            push (first, &s2);
73
74         }
75      }
76
77      while (!isEmpty (s2)) {
78         push (pop (&s2), s1);
79      }
80
81  }
```

```
Original Stack:
20
10
4
3


Changed Stack of s2:
4
3
20
10


Changes Stack of s1:
10
20
3
4
```

Question 4: Use pop and push operations to simulate the behavior of **dequeue** operation of a queue. Note that enqueue adds an element to the rear end of the queue and dequeue removes an element from the front of the queue.For example, **enqueue** can be written as:

```
void enqueue (int element, Stack * s) {
     push (element, &s);
 }

int dequeue (Stack * s) {
     // write your answer here
}
```

```
Stack tempStack;

while(!isempty(*s)) {
   push(pop(&s), &tempStack);
}

int e = pop(&tempStack);

while(!isempty(*tempStack)) {
   push(pop(&tempStack), &s);
}
```