Shoubra Faculty of Engineering
Benha University

Computer Engineering Department
3rd year

Data Structure: Assignment #4

Programming problems:

1. Write a method - ***String decimalToBinary (int num)*** - that prints out the binary equivalent of a decimal integer using the stack implementation. For example, if the number 44 is input, the program will print out 101100. In addition, add a method - ***String decimalToBase (int num,int base)*** - that could print the equivalent of a decimal integer for a specified user defined base (i.e. 2 -> 16).

   Soln

```
String decimalToBinary(int num)
   {
       String str="";
      int n=num;
       int rem;
       LinkStack stk1 = new LinkStack();
       while(num>0)
       {
            rem = num%2;
            stk1.push(rem);
            num /=2;
       }
       while(!stk1.isEmpty())
       {
           str = str + stk1.pop();
       }

       return ("The Binary Number of ("+n+") is: "+ str);
   }
```

```
String decimalToBase(int num,int base)
    {
        String str="";
        int n=num;         int rem;
        LinkStack stk1 = new LinkStack();
        char [] digits = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
if( base>=2 && base <=16)  {
        while(num>0)
        {
             rem = num%base;
             stk1.push(rem);
             num /=base;
        }
        while(!stk1.isEmpty())
        {
            str = str + digits [ stk1.pop()];
        }
       str= "The Number ("+n+")in base "+ base +" is: "+ str;
}
else
{
str= "Not Allowed Base";
}


        return str;    }
```

2. Create a data structure TwoStacks class that represents two stacks using only one array, i.e., both two stacks should use the same array for storing elements. Following functions must be supported by TwoStacks.
   push1(int** x**) –> pushes x to first stack
   push2(int** x**) –> pushes x to second stack
   pop1( ) –> pops an element from first stack and return the popped element
   pop2 ( ) –> pops an element from second stack and return the popped element

Note: Implementation of TwoStacks should be space efficient.

Soln

```
public class TwoStacks
{
        int size;
    int top1, top2;
    int arr[];
    // Constructor
    TwoStacks(int n)
    {       arr = new int[n];
        size = n;
        top1 = -1;
        top2 = size;    }
    // Method to push an element x to stack1
    void push1(int x)
    {       // There is at least one empty space for new element
        if(top1 < top2 - 1)
        {
            top1++;
            arr[top1] = x;
        }
        else
        {
            System.out.println("Stack Overflow");
            System.exit(1);
        }
    }
```

```
// Method to push an element x to stack2
void push2(int x)
{
  // There is at least one empty space for
  // new element
  if(top1 < top2 -1)
  {
    top2--;
    arr[top2] = x;
  }
  else
  {
    System.out.println("Stack Overflow");
    System.exit(1);
  }
}
// Method to pop an element from first stack
int pop1()
{
  if(top1 >= 0)
  {
    int x = arr[top1];
    top1--;
    return x;
  }
```

```
        else
        {
            System.out.println("Stack Underflow");
            System.exit(1);
        }
        return 0;
    }
    // Method to pop an element from second stack
    int pop2()
    {
        if(top2 < size)
        {
            int x =arr[top2];
            top2++;
            return x;
        }
        else
        {
            System.out.println("Stack Underflow");
            System.exit(1);        }
        return 0;
    }
}
```