**Algorithms**

Pseudocode

**Directions**

This assignment is broken into two sections, the first which is a series of small programs you will create using the IB standard for pseudocode, and the second of which contains actual IB exam questions. In order to answer all pseudocode based questions you should use the IB compiler that we saw in class (<http://ibcomp.fis.edu/pseudocode/pcode.html>) and then copy and paste (or screenshot) your finished code into this document.

**Section 1 Questions**

1. In this program you will ask the user to input an integer number and then produce the binary value of that number. The problem is, pseudocode doesn’t allow you to cast numbers to integers, but it does allow you to use **mod** to do modulus (15 mod 2, for example).
   1. Let’s say the user inputs a number 15. Give an output statement which

produces both the integer value 7 and the remainder of 1:

|  |
| --- |
| num=15  remain=num mod 2  Div=(num-remain)/2  output(Div)  output(remain) |

* 1. Create a loop which will print out the binary number for any given input:

|  |
| --- |
| num=15  loop i from 0 to 7  remain=num mod 2  num=(num-remain)/2  output remain  end loop |

* 1. Create an array that can store an 8-bit binary number and then modify your code so that it stores the binary number produced in b. For this case you may assume the input is always below 256:

|  |
| --- |
| Binary = [0,0,0,0,0,0,0,0]  num=15  loop i from 0 to 7  remain=num mod 2  num=(num-remain)/2  Binary[i]=remain  end loop  loop j from 0 to 7  output Binary[j]  end loop |

1. A sequence is described by a set of simple rules. Begin with an integer and then repeat, following these rules:

**1: If the number is even, divide by 2**

**2: If the number is odd, multiply 3 and then add 1**

**3: If the number is 1, then stop**

* 1. Create a loop which prints out the values of the sequence,

|  |
| --- |
| num = 15  loop while num > 1  if num mod 2 = 0 then  num = num / 2  else  num = (num\*3) + 1  end if  output num  end while |

* 1. Given that you don’t know how long a sequence will be, storing the values in an array is not feasible. Instead, use a collection (see **Names Collection** in drop down on the IB compiler page**)** to store all the values and then print them out at the end.

|  |
| --- |
| numsequence = new Collection()  num = 15  loop while num > 1  if num mod 2 = 0 then  num = num / 2  else  num = (num\*3) + 1  end if  numsequence.addItem(num)  end while  numsequence.resetNext()  loop while numsequence.hasNext()  output numsequence.getNext()  end while |

* 1. Create an array that contains all the numbers in your collection that are multiples of 5. See the **Prime Array** demo program as an example of creating arrays.

|  |
| --- |
| numMulti5 = new Array()  num = 15  N=0  loop while num > 1  if num mod 2 = 0 then  num = num / 2  else  num = (num\*3) + 1  end if  if num mod 5 = 0 then  numMulti5[N] = num  else  numMulti5[N] = 0  end if  N=N+1  end while  loop i from 0 to 30  if numMulti5[i] > 0 then  output numMulti5[i]  end if  end loop |

1. You will create two different arrays of 4 names which are at least 4 letters in length and use them to create mashed up names.
   1. Using the **Cities Array** demo as an example, loop through and printing out the first and last three letters of each name by creating methods called **firstThree** and **lastThree.** You should print out the first and last three from each of the lists. That means in the first time through the loop print out the the first name from both lists.

|  |
| --- |
| name = [names, names2]  names = ["Dallas","Denver","Yorker","Romsle"]  names2 = ["Blaecs,"Monous", "Porker", "Lsomer"]  loop i from 0 to 4  loop j from 0 to 1  output FirstThree(name[j][i])  output LastThree(name[j][i])  end loop  end loop    method FirstThree(s)  return s.substring(0,3)  end method  method LastThree(s)  return s.substring(3,6)  end method |

* 1. Modify the code from part **(a)** so that it stores these first and last names three in a stack (see **Stack Reverse List** for an example). Push them one at a time in any order that you like.

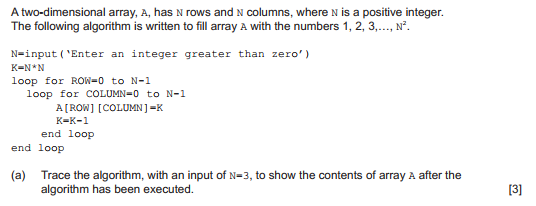
|  |
| --- |
| name = [names, names2]  STACK = new Stack()  names = ["Dallas","Denver","Yorker","Romsle"]  names2 = ["Blaecs,"Monous", "Porker", "Lsomer"]  loop i from 0 to 4  loop j from 0 to 1  output FirstThree(name[j][i])  output LastThree(name[j][i])  STACK.push(names[COUNT])  STACK.push(names2[COUNT])  end loop  end loop    method FirstThree(s)  return s.substring(0,3)  end method  method LastThree(s)  return s.substring(3,6)  end method  loop while NOT(STACK.isEmpty())  NAME = STACK.pop()  NAME2 = STACK.pop()  output NAME + NAME2  end loop |

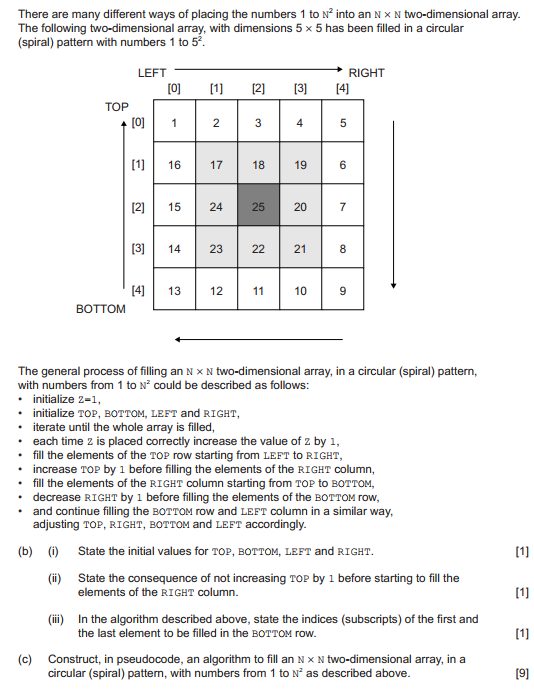
* 1. In a final loop pop items off the stack and create new mashed up names which are all six letters. Print them off and also add them to a collection.

|  |
| --- |
| name = [names, names2]  STACK = new Stack()  names = ["Dallas","Denver","Yorker","Romsle"]  names2 = ["Blaecs,"Monous", "Porker", "Lsomer"]  loop i from 0 to 4  loop j from 0 to 1  output FirstThree(name[j][i])  output LastThree(name[j][i])  STACK.push(names[FirstThree(name[j][i])])  STACK.push(names2[LastThree(name[j][i])])  end loop  end loop    method FirstThree(s)  return s.substring(0,3)  end method  method LastThree(s)  return s.substring(3,6)  end method  loop while NOT(STACK.isEmpty())  NAME = STACK.pop()  NAME2 = STACK.pop()  output NAME + NAME2  end loop |

**Section 2 Questions**

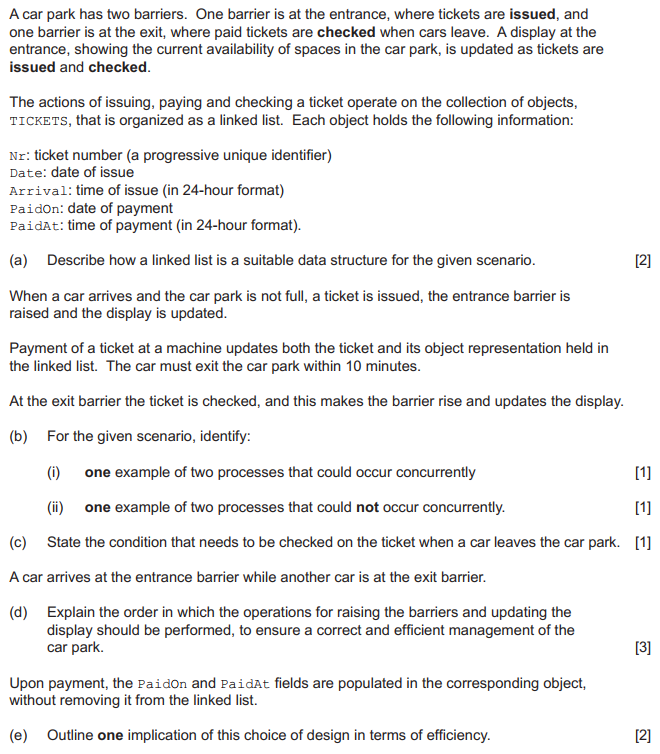
1. 2D Arrays (N16 - 14)





|  |  |
| --- | --- |
| 1. A=[[9,8,7], [6,5,4],[3,2,1]]    1. TOP=1 BOTTOM=13 LEFT=1 RIGHT=5    2. It would overwrite the right corner that was initialized in the beginning.    3. 13,9  |  | | --- | | N = 5  r=N-1  r0=0  In=1  Grid=N\*N  GridMap=[]  x=0  y=0  LOOP i FROM 0 TO N  GridMap.append([])  LOOP i FROM 0 TO N  GridMap[i].append(0)  END LOOP  END LOOP  LOOP WHILE In<Grid:  LOOP WHILE x!=r:  GridMap[y][x]=In  In+=1  x+=1  END WHILE  LOOP WHILE x==r and y!=r:  GridMap[y][x]=In  In+=1  y+=1  END WHILE  LOOP WHILE x!=r0 and y==r:  GridMap[y][x]=In  In+=1  x-=1  END WHILE  LOOP WHILE y!=r0 and x==r0  GridMap[y][x]=In  y-=1  In+=1  END WHILE  IF (r-1)!=0:  r-=1  END IF  IF (r0+1)!=N-1 THEN  r0+=1  END IF  y+=1  x+=1  END WHILE    GridMap[y][x]=In  LOOP i FROM 0 TO len(GridMap)):  OUTPUT(GridMap[i])  END LOOP | |

2) Generalized Algorithms (M16 - 13)



(continued on next page)

