**BBM201 – Data Structures – Fall 2017**

**1st Midterm**

**02.11.2017 – 110 minutes**

**Name Surname:**

**Student ID :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Questions | 1 | 2 | 3 | 4 | 5 | 6 | Total |
| Points | 15 | 16 | 14 | 14 | 21 | 20 | 100 |
| Grade |  |  |  |  |  |  |  |

**1. (Recursion) *(15 points)***

a. Write the output of the following code. ***(5 points)***

|  |
| --- |
| **void** **mystery**(**int** n){  **if**(n==0) **return**;  **for**(**int** i=0;i<n;i++){  **printf**("%d", n);  mystery(n-1);  }  }  **int** **main**()  {  mystery(2);  **return** 0;  } |

b. Please write the recursive method that applies exponentiation on a given integer x, so it returns xm. Global variables are not allowed for the solution. ***(10 points)***

|  |
| --- |
| **int** **exp**(**int** x, **int** m) {  } |

**2.** **(Stack/Queue)** Given the initial empty position of stack and queue (circular) below, give the final representation of data below for array representations and fill the values of top, front and rear positions into an array of size 8. ***(16 points)***

a. Stack ***(8 points)***

Push (5) , Push (2) , Pop () , Push (6) , Push (3) , Pop () , Push (8) , Push (2) , Push (7) , Push (4) , Push (6) , Push (3) , Push (8)

0 1 2 3 4 5 6 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
|  | Top |
| Initial | -1 |
| Final |  |

b. Circular Queue ***(8 points)***

Enqueue(5) , Enqueue(2) , Dequeue () , Enqueue(6) , Enqueue(3) , Dequeue () , Enqueue(8) , Enqueue(2) , Enqueue(7) , Enqueue(4) , Enqueue(1)

0 1 2 3 4 5 6 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | First | Rear |
| Initial | -1 | -1 |
| Final |  |  |

**3.** **(Pointers)** If C is the array shown with its address above each node, write what the following lines of a program will print in the empty column. ***(14 points)***

116

132

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 1 | 5 | 0 | 2 | 9 | 8 | 7 | 2 | 10 | 12 |

int C[3][2][2];

|  |  |
| --- | --- |
| The code | will print: |
| printf("C+1=%d", C+1); |  |
| printf("\*(C+2)=%d", \*(C+2)); |  |
| printf("\*(\*(C+2)+1)) =%d", \*(\*(C+2)+1)); |  |
| printf(" \*(\*\*C+1)=%d ’’, \*(\*\*C+1)) ; |  |
| printf(" \*(C[2][1]+1)=%d ’’, \*(C[2][1]+1)); |  |
| printf(" \*(\*((\*(C+1))+1))=%d ’’, \*(\*((\*(C+1))+1)); |  |
| printf("\*(\*(C+2)+2)) =%d", \*(\*(C+2)+2)); |  |

**4.** **(Performance)** Please give time complexities of the functions given below, compare them in a decreasing order. ***(14 points)***

1. 5n2 – 6n = O ( )
2. 2n2 + nlogn = O ( )
3. n3 + 106n2 =O ( )
4. nk + n + nklogn = O ( )
5. for (i=0; i<n; i++){ O ( )

      for (j=0; j<=n-1; j++){

         for (k=j; k<=n-1; k++){

            … *loop body*…

          }

      }

   }

1. for (i=0; i<n; i++){ O ( )

        for (j=1; j<=n-1; j\*2){

        … *loop body* …

        }

     }

Order =

**5.** **(Problem Solving)** Write a C function to find out the maximum and second maximum numbers from a given two-dimensional array of integers. Complete the code below where there are spaces “\_\_\_\_\_\_” as needed and inside the function TwoMax. Your answer should trace the array only once for full credit. ***(21 points)***

|  |
| --- |
| **int main()**  **{**  int first=0, second=0;  int \*\*x;  //allocate for the two-dimensional array  x = malloc(rows \* sizeof ( \_int \*\_\_\_\_\_\_));  for (int i=0; i<rows; i++)  {  x[i] = malloc(cols \* sizeof( int ));  }  initData(x);  //assume this function fills the array x with random values  TwoMax( \_\_\_&\_\_ x, rows, cols, \_&\_\_first, \_&\_\_second)  printf(“The maximum number is %d, second maximum is %d\n”, first, second);  return 1;  **}**  // arr is two-dimensional array,  //n is the first dimension of the array,  //m is the second dimension of the array  **int TwoMax( int \_\_\*\*\*\_\_ arr, int n, int m, int \_\*\_\_first, int \_\_\*\_second)**  **{**  **}** |

**6.** **(Sparse Matrix)** A matrix is called symmetric if for all values of i and j satisfies A[i][j] = A[j][i]. A sparse matrix has at least + 1 zero values out of all m items in the matrix. Given that a matrix A is sparse, symmetric and square (NxN) ***(20 points)***

a. Propose and describe an efficient representation to improve the space complexity compared to two-dimensional representation of matrix A. ***(7 points)***

b. Justify and show mathematically your proposed representation is space efficient. ***(7 points)***

c. Given the two-dimensional representation of A, describe and give pseudocode of converting two-dimensional representation into your proposed representation using minimum number of accesses to matrix A. Give the complexity of this operation in tilde (~) notation (Note: this part of the question looks for the most efficient approach for full credit) ***(6 points)***