1. (b) Test 1

Test Summary

- No. of Sections: 2
- No. of Questions: 3
- Total Duration: 45 min

Section 1 - Coding Proficiency

Section Summary

- No. of Questions: 2
- Duration: 30 min

Additional Instructions:

None

Q1. **Compete Cell**

There is a colony of 8 cells arranged in a straight line where each day every cell competes with its adjacent cells(neighbour). Each day, for each cell, if its neighbours are both active or both inactive, the cell becomes inactive the next day, otherwise it becomes active the next day.
Assumptions: The two cells on the ends have single adjacent cell, so the other adjacent cell can be assumed to be always inactive. Even after updating the cell state. consider its previous state for updating the state of other cells. Update the cell information of all cells simultaneously. Write a function cellCompete which takes takes one 8 element array of integers cells representing the current state of 8 cells and one integer days representing the number of days to simulate. An integer value of 1 represents an active cell and value of 0 represents an inactive cell.

Input Format

Input will have 8 array values and the no of days

Output Format

print the array

Constraints

array size is 8 integers

Sample Input

```
1 0 0 0 0 1 0 0
1
```

Sample Output

```
0 1 0 0 1 0 1 0
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. **Least Recently Used**

The LeastRecentlyUsed(LRU) cache algorithm exists the element from the cache(when it's full) that was leastrecentlyused. After an element is requested from the cache, it should be added to the cache(if not already there) and considered the most recently used element in the cache.

Initially, the cache is empty. The input to the function LruCountMiss shall consist of an integer max_cache_size, an array pages and its length len. The function should return an integer for the number of cache misses using the LRU cache algorithm. Assume that the array pages always has pages numbered from 1 to 50
*int lruCountMiss(int max_cache_size, int *pages,int len)*
{/ /write tour code }

Input Format

Input consists of an integer max_cache_size, array length len and an array pages

Output Format

an integer for the number of cache misses using the LRU cache algorithm

Constraints

Should write a function

Sample Input

Sample Output

3 16 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0	11

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Section 2 - Essay Writing

Section Summary

- No. of Questions: 1
- Duration: 15 min

Additional Instructions:

None

Q1. **ESSAY WRITING**

Write a response that describe the traits of a good leader. To what extend do you agree with the statement? Explore arguments both for and against the statement in your response.

Directions

“He who has never learned to obey cannot be a good leader.”

Keywords

Answer Key & Solution

Section 1 - Coding Proficiency

Q1

Test Case

Input

```
0 1 0 1 0 1 1 0
3
```

Output

```
1 0 1 1 1 1 0 0
```

Weightage - 10

Input

```
0 0 0 0 1 1 1 1
5
```

Output

```
1 0 0 1 0 1 0 1
```

Weightage - 10

Input

```
0 0 0 1 0 0 0 0
1
```

Output

```
0 0 1 0 1 0 0 0
```

Weightage - 10

Input

```
1 1 1 1 0 1 1 1
2
```

Output

```
0 1 1 0 0 0 0 0
```

Weightage - 10

Input

```
1 1 0 0 0 0 1 0
10
```

Output

```
0 0 1 1 0 1 0 0
```

Weightage - 10

Input

```
0 0 0 0 1 1 1 1
100
```

Output

```
0 0 1 1 1 1 1 0
```

Weightage - 10

Input

```
1 1 1 1 1 0 0 0
1000
```

Output

```
1 0 1 0 0 1 1 0
```

Weightage - 10

Input

```
1 0 0 1 1 0 1 1
864
```

Output

```
1 1 0 0 1 1 1 1
```

Weightage - 10

Input

```
0 0 1 1 0 0 1 1
723
```

Output

```
1 0 0 0 0 0 1 1
```

Weightage - 10

Input

```
0 0 1 0 1 0 1 0
123
```

Output

```
0 0 0 0 0 1 0 0
```

Weightage - 10

Sample Input

```
1 0 0 0 0 1 0 0
1
```

Sample Output

```
0 1 0 0 1 0 1 0
```

Solution

```
#include<stdio.h>
int* cellCompete( int* , int , int);
void display( int* , int);
int main()
{
int arr[8], size = 8 , days, index;
for(index=0 ; index<8 ; index++)
    scanf("%d",&arr[index]);
scanf("%d",&days);
cellCompete( arr , size , days);
display( arr , size);
return 0;
}
void display(  int* arr , int size)
{
    int ctr;
    for( ctr = 0 ; ctr < size ; ctr++)
        printf("%d " , arr[ctr]);
}
int* cellCompete( int* arr , int size , int days)
{
    int ctr ,prev , nextprev ;
while( days)
{
    prev = 0;
    for( ctr = 0 ; ctr < size-1 ; ctr++)
    {
```

```
#include<stdio.h>
int* cellCompete( int* , int , int);
void display( int* , int);
int main()
{
int arr[8], size = 8 , days, index;
for(index=0 ; index<8 ; index++)
    scanf("%d",&arr[index]);
scanf("%d",&days);
cellCompete( arr , size , days);
display( arr , size);
return 0;
}
void display(  int* arr , int size)
{
    int ctr;
    for( ctr = 0 ; ctr < size ; ctr++)
        printf("%d " , arr[ctr]);
}
int* cellCompete( int* arr , int size , int days)
{
    int ctr ,prev , nextprev ;
while( days)
{
    prev = 0;
    for( ctr = 0 ; ctr < size-1 ; ctr++)
    {
```

```
            nextprev = arr[ctr];
            arr[ctr] = prev ^ arr[ctr+1];
            prev = nextprev;
        }
        arr[ctr] = prev ^ 0;
        days--;
    }
    return arr;
    }
```

```
            nextprev = arr[ctr];
            arr[ctr] = prev ^ arr[ctr+1];
            prev = nextprev;
        }
        arr[ctr] = prev ^ 0;
        days--;
    }
    return arr;
    }
```

Q2

Test Case

Input

```
2   9
2  3  1  3  2  1  4  3  2
```

Output

```
8
```

Weightage - 5

Input

```
9  10
9  30  36  5  3  28  5  46  19  26
```

Output

```
9
```

Weightage - 5

Input

```
13  11
0  39  41  17  42  44  3  14  21  14  46
```

Output

```
10
```

Weightage - 5

Input

```
6  42
3  16  29  16  14  19  21  18  46  37  7  21  2  45  39  10  45
```

Output

```
38
```

Weightage - 10

Input

```
17  38
23  0  11  31  38  8  26  4  38  49  20  23  4  7  42  33  3  14
```

Output

```
31
```

Weightage - 10

Input

```
23  28
23  13  14  6  36  16  15  33  4  16  1  16  2  38  33  35  2  4
```

Output

```
17
```

Weightage - 5

Input

```
8 35
42 7 0 42 21 48 23 41 7 44 44 22 37 49 15 37 20
```

Output

```
31
```

Weightage - 10

Input

```
27 49
41 15 20 46 44 16 34 33 12 23 49 0 16 14 13 34
```

Output

```
28
```

Weightage - 10

Input

```
8 22
11 42 6 20 29 42 46 17 26 15 39 0 10 13 48 36 4
```

Output

```
21
```

Weightage - 5

Input

```
12 17
20 16 32 21 46 10 18 19 45 36 39 2 11 46 15 16
```

Output

```
16
```

Weightage - 5

Input

```
19 47
28 11 48 26 41 17 37 47 10 6 45 0 22 26 28 48 4
```

Output

```
33
```

Weightage - 10

Input

```
40 46
7 48 15 23 16 48 39 30 26 42 16 15 30 39 32 8 1
```

Output

```
33
```

Weightage - 20

Sample Input

```
3   16
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0
```

Sample Output

```
11
```

Solution

```
#include<stdio.h>
#include<limits.h>
#include<malloc.h>
int isHit(int **arr , int r , int c , int page);
int findMin( int**arr , int r , int c);
void init( int** arr , int r , int c);
int main()
{
int *input, noe, index;
int ctr , cache_size ,miss = 0 , find ,min_ind;
int **arr;
scanf("%d",&cache_size);
scanf("%d",&noe);
input = (int*)malloc(sizeof(int)*noe);
for(index=0 ; index<noe ; index++)
    scanf("%d",&input[index]);
// allocate memory
arr = (int**) malloc( cache_size * sizeof(int*));
for(ctr = 0 ; ctr < cache_size; ctr++)
    arr[ctr] = (int*)malloc( sizeof(int) * 2);
// initialize an array with -1
init(arr , cache_size , 2);

    for( ctr = 0 ; ctr < noe ; ctr++)
    {
        find = isHit( arr , cache_size,2,input[ctr]);
        if( find  == 0)
        {
            min_ind = findMin(arr , cache_size , 2);
            arr[min_ind][0]=ctr;
            arr[min_ind][1]=input[ctr];
            miss++;
        }
        else
        {
            arr[find][0] = ctr;
        }
    }
printf("%d" , miss);
return 0;
}
int isHit(int **arr , int r , int c , int page)
{
    int row;
    for( row = 0 ; row < r ; row++)
    {
        if( arr[row][1] == page)
             return row;
    }
return 0;
}
int findMin( int**arr , int r , int c)
{
    int row , min =INT_MAX , minpos;
    for( row = 0 ; row < r ; row++)
    {
        if( arr[row][0] < min)
        {
            min = arr[row][0];
            minpos = row;
        }

    }
    return minpos;
}
void init( int** arr , int r , int c)
```

```
{
    int row , col;
    for(row = 0 ; row < r ; row++)
   {
     for( col = 0 ; col < c; col++)
         arr[row][col] = -1;
   }
}



#include<stdio.h>
#include<limits.h>
#include<malloc.h>
int isHit(int **arr , int r , int c , int page);
int findMin( int**arr , int r , int c);
void init( int** arr , int r , int c);
int main()
{
int *input, noe, index;
int ctr , cache_size ,miss = 0 , find ,min_ind;
int **arr;
scanf("%d",&cache_size);
scanf("%d",&noe);
input = (int*)malloc(sizeof(int)*noe);
for(index=0 ; index<noe ; index++)
    scanf("%d",&input[index]);
// allocate memory
arr = (int**) malloc( cache_size * sizeof(int*));
for(ctr = 0 ; ctr < cache_size; ctr++)
    arr[ctr] = (int*)malloc( sizeof(int) * 2);
// initialize an array with -1
init(arr , cache_size , 2);

    for( ctr = 0 ; ctr < noe ; ctr++)
    {
        find = isHit( arr , cache_size,2,input[ctr]);
        if( find  == 0)
        {
            min_ind = findMin(arr , cache_size , 2);
            arr[min_ind][0]=ctr;
            arr[min_ind][1]=input[ctr];
            miss++;
        }
        else
        {
            arr[find][0] = ctr;
        }
    }
printf("%d" , miss);
return 0;
}
int isHit(int **arr , int r , int c , int page)
{
    int row;
    for( row = 0 ; row < r ; row++)
    {
        if( arr[row][1] == page)
             return row;
    }
return 0;
}
int findMin( int**arr , int r , int c)
{
    int row , min =INT_MAX , minpos;
```

```
    for( row = 0 ; row < r ; row++)
    {
        if( arr[row][0] < min)
        {
            min = arr[row][0];
            minpos = row;
        }

    }
    return minpos;
}
void init( int** arr , int r , int c)
{
    int row , col;
    for(row = 0 ; row < r ; row++)
   {
     for( col = 0 ; col < c; col++)
         arr[row][col] = -1;
   }
}
```

Section 2 - Essay Writing

Q1

Sample Essay

No Essay

Keywords

GOOD, LEADER, OBEY, NEVER, LEARNED, CANNOT,