CMPE-231 DATA STRUCTURES

••• Laboratory Work # 5

5

Duration – 100 minutes

Recursive Programming and Queues Data structure.

✧ Experiment 1 *(PreLab task – to be prepared in advance)*

(to write a series of small complete C programs required by the task 3.1.3 in the textbook, page 126)

REMINDER:

- Every recursive function must have one or more base cases,
  - The general (recursive) case must eventually be reduced to a *base case* (terminating condition),
  - The base case stops the recursion,
  - Every recursive call has its own copy of parameters and the local variables [1],
  - A *recursive function* is a function that calls itself [5].

Let `a` be an array of integers. Present recursive algorithms and write their C implementations to compute:

- the maximum (minimum) element of the array,
- the sum (product) of the elements of the array,
- the average of the elements of the array

HINT: Use functions having the following prototypes:

```
<return type> name_of_the_function(int a[ ], int n);
```

For example, function that computes maximum/minimum element in the array can be named as `find_max( )` / `find_min( ).`

«The concept of recursion is fundamental in mathematics and computer science. The simple definition is that a recursive program in a programming language is one that calls itself (just as a recursive function in mathematics is one that is defined in terms of itself). … All practical computations can be couched in a recursive framework. … Many interesting algorithms are simply expressed with recursive programs, and many algorithm designers prefer to express methods recursively».

Source: R.Sedgewick. *Algorithms in C*. Parts 1-4 (Fundamentals, Data Structures, Sorting, Searching), 3rd edition, Addison-Wesley Publ., 1998, 702 p.

«There are two approaches to writing repetitive algorithms: iteration and recursion. … You should not use recursion if the answer to any of the following questions is **"no"**:

1. Is the algorithm or data structure naturally suited to recursion?
2. Is the recursive solution shorter and more understandable?
3. Does the recursive solution run within acceptable time and space limits?».

Source: R.F.Gilberg, B.A.Forouzan. *Data Structures. A Pseudocode Approach with C*, 2nd edition, Course Technology (Thomson), 2005, 720 p.

HOMEWORK

Experiment 2 *(PreLab task – to be prepared in advance)*

Consider the following recursive algorithm (pseudocode)

```
algorithm fun1(x)
if(x < 5)   return (3 * x)
else   return (2 * fun1(x - 5) + 7)
end if
end fun1
```

What would be returned if **fun1** is called as **fun1(12)**? Perform analysis of the problem manually, without using computer.

Experiment 3 – *to be prepared andExplained by an assiatant*

3.1) The following programs can be used for self-studying for main operations(Insertions and Deletions of integer values)) of Circular Queues: Run and discuss the output.

```
/*  ************************************  */
/*  Queues main  Operations                */
/*  Inserting & Deleting Integer values into/from  */
/*     a Circular queue                    */
/*  Insertion(I) and Deletion(D) Operationions     */
/*   End Of operation(E)                   */
/* ************************************** */
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

#define MAXELEMENTS 5
#define TRUE 1
#define FALSE 0
// circular Queue insertion and deletion functions

struct queue
  {
   int items[MAXELEMENTS];
   int  front , rear ;
  } ;

 void cqinsert(struct queue * , int);
 int cqdelete(struct queue *);
 int empty(struct queue *);

 int main(void)
 {
 char operation;
 int x;
 struct  queue q;
 q.front = q.rear = MAXELEMENTS - 1;
 do
 {
  printf("%s\n","Insert Operation type I  D or E ");
  scanf("\n%c",&operation);
   switch (operation)
```

```
  {
   case 'I':printf("%s\n","Insert an element");
        scanf("\n%d",&x);
        cqinsert(&q , x);
        break;
   case 'D':x=cqdelete(&q);
    printf("\n %d is deleted \n",x);
    break;
   }
 }
while (operation != 'E');
return 0;
}

int empty(struct queue *pq)
{
return((pq->front == pq->rear) ? TRUE : FALSE);
}

int cqdelete(struct queue *pq)
{
if (empty(pq)) {
          printf("Queue underflow ");
          exit(1);
          }
if (pq->front == MAXELEMENTS - 1)
                       pq->front = 0;
                else
           (pq->front)++;
return(pq->items[pq->front]);
}

void cqinsert(struct queue *pq , int x)
{
/* make room for new element */

if (pq->rear == MAXELEMENTS - 1)
                           pq->rear = 0;
                else
                (pq->rear)++;
if (pq->rear == pq->front) {
                    printf("Queue  overflow");
                          getchar(); getchar();
                    exit(1);
                    }
printf("\n %d is inserted %d\n",x);
pq->items[pq->rear] = x;
}
```

3.2) Do some modifications in the previous program and display the insertion and Deletion values and location points of the Circular Queue structure.

Experiment 4 *(PreLab task–to be prepared in advance by the student)*

(to write a COMPLETE C program)

Prepare a menu driven C program for Inserting and deletion operation for a queue structure of employee information which has the following structure declaration. Structure of a queue will be defined as follows.

```
struct person
{
        int empNo;
        char name[12];
        int age;
        char gender[2];/* M=male, F=Female */
};
struct queue
{
  struct person allperson[15];
  int  front , rear;
};
```

Define the following Initialized array of structure in your program.

```
sruct person tenPerson[10]={123,"Ahmet", 21, "M",
      234,"Sevgi", 26, "F", 128,"Osman",18,"M", 432,"Mert",27,"M",
      287,"Ayse",34,"F"   ,  423,"Kemal", 21, "M",634,"Lale", 16, "F",
      828,"Sefer",15,"M",   252,"Meral",27,"F", 887,"Demet",34,"F"};
```

Following steps of operations will take place in your menu driven program.

1. Create a Circular queue using **tenPerson** array structure(copy from array into queue will be done).
2. Delete all the elements of queue and list all the deleted from the monitor.
3. Using circular queue which is populated at step 1, Create two new circular queues, one for Male and one for Female employees. These new queues will be created during the deletion of circular queue and use gender fields for determining Male(M) and Female(F) queues.
4. List the content of Male queue and Female queue during delete operation of each queue.
5. End of operation