Tutorial Sheet – ODD 2021
Week 4 :
Topic : Recursion and Recursion removal using stack, Median search

Q1) Consider the following recursive function:

```
int mystery(int number)
{ if (number == 0)
return number;
else return(number + mystery(number - 1)); }
```

a. Identify the base case.
b. Identify the general case.
c. What valid values can be passed as parameters to the function mystery?
d. If mystery(0) is a valid call, what is its value? If not, explain why.
e. If mystery(5) is a valid call, what is its value? If not, explain why.
f. If mystery(-3) is a valid call, what is its value? If not, explain why.

2) Write a recursive algorithm to multiply two positive integers m and n using repeated addition. Specify the base case and the recursive case.

Q3. Given a sorted array with possibly duplicate elements,rite an efficient code to find the first and last position of the repeated element in the array. If element is found then code should return the index. If element not found then code should -1,-1;

For example:

Example 1:

Array A= [5, 7, 7, 8, 8, 10] Key= 8 Output= [3, 4]

Array A= [5, 7, 7, 8, 8, 10] Key= 6 Output= [-1, -1]

Solution

Q1)

a. The statements in Lines 3 and 4.
b. The statements in Lines 5 and 6.
c. Any nonnegative integer.
d. It is a valid call. The value of mystery(0) is 0.
e. It is a valid call. The value of mystery(5) is 15.

f. It is an invalid call. It will result in an infinite recursion.

Q2)

$$multiply(m,n) = \begin{cases} 0 & if\ n = 0 \\ m & if\ n = 1 \\ m + multiply(m, n-1) & otherwise \end{cases}$$

Q3) **Naïve Approach**

1. Run a for loop and for i = 0 to n-1
2. Take first = -1 and last = -1
3. When we find element first time then we update first = i
4. We always update last=i whenever we find the element.
5. We print first and last.

Efficient Approach

1. For the first occurrence of a number

a) If (high >= low)

b) Calculate mid = low + (high - low)/2;

c) If ((mid == 0 || x >arr[mid-1]) &&arr[mid] == x)

return mid;

d) Else if (x >arr[mid])

return first(arr, (mid + 1), high, x, n);

e) Else

return first(arr, low, (mid -1), x, n);

f) Otherwise return -1;

2. For the last occurrence of a number

a) if (high >= low)

b) calculate mid = low + (high - low)/2;

c)if((mid == n-1 || x <arr[mid+1]) &&arr[mid] == x)

return mid;

d) else if(x <arr[mid])

return last(arr, low, (mid -1), x, n);

e) else

return last(arr, (mid + 1), high, x, n);

f) otherwise return -1;

5)

Recursive function

```
Node* insertEnd(Node* head, int data)

{
    // If linked list is empty, create a
    // new node (Assuming newNode() allocates
    // a new node with given data)
    if (head == NULL)
         return newNode(data);

    // If we have not reached end, keep traversing
    // recursively.
    else
        head->next = insertEnd(head->next, data);
    return head;
}
```