PDPM IIITDM JABALPUR
ES103 DATASTRUCTURES AND ALGORITHMS
SOLUTION: QUIZ 1 (February 09, 2016)

MaxMarks 36 **Time 1 hour**

1. [1] The minimum number of arithmetic operations required to evaluate the polynomial

$$P(X) = X^5 + 4X^3 + 6X + 5$$

for a given value of X. How would you get that?

Answer: 7 (4 multiplication and 3 addition)

We can parenthesize the polynomial to minimize the number of operations (See Horner's Method). We get $X(X^2(X^2 + 4) + 6) + 5$ after parenthesization.

2. [1] Consider the following problem, which is known as Josephus Problem.

The problem postulates a group of soldiers surrounded by an overwhelming enemy force. There is no hope for victory without reinforcements, but there is only a single horse available for escape. The soldiers agree to a pact to determine which of them is to escape and summon help. They form a circle and a number N is picked from a hat. One of their names is also picked from a hat. Beginning with the soldier whose name is picked, they begin to count clockwise around the circle. When the count reaches N, that soldier is removed from the circle, and the count begins again with the next soldier. The process continues so that each time the count reaches N, another soldier is removed from the circle. Any soldier removed from the circle is no longer counted. The last soldier remaining is to take the horse and escapes.

Which data structure would you find suitable to solve this problem?

Answer: Circular linked list

3. [2] A program P reads in 1000 integers in the range [0,100] representing the scores of 1000 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies?

(a) An array of __________ numbers

(d) A dynamically allocated array of _____ numbers

Answer: Option (a) An array of 50 numbers

We have to store frequencies of scores above 50. That is number of students having score 51, number of students having score 52 and so on. For that an array of size 50 is the best option.

4. [2] What is the running time of the following function in Θ-notation. Show the steps in the right side of the code:

```
int FuncSum(int n)

{

  int i, j, k = 0;
  for (i = n/2; i<=n; i++)
            for (j = 2; j<=n; j = j * 2)
                      k=k+n/2;

 return k;

}
```

Answer: The outer loop runs n/2 or Θ(n) times. The inner loop runs Θ(logn) times (Note that j is divide by 2 in every iteration). So the statement "k = k + n/2;" runs Θ(nlogn) times.

5. [5] Write the worst case complexity of the following operations:

(a) Searching a sorted array of length n for a given element. **O(n)**

(b) Searching an unsorted doubly linked list for a given element. **O(n)**

(c) Merging two linked lists of size m and n, respectively. **O(m+n)**

Find middle element of the linked list:

(d) Method 1: For each of the node count how many nodes are there in the list and see whether it is the middle. **O(n^2)**

(e) Method 2: Traverse the list and find the length of the list. Again scan the list and locate length/2 node from the beginning. **O(n)**

6. [2] Calculate the asymptotic complexity in Big-O notation of the following fragments of the code. Show the steps in the right side of the code:

```
int i, count = 0;
        for (i=1; i*i<=n; i++)
          count++;
```

Answer: The loop will run if $i^2 \le n$. This gives run-time O($\sqrt{n}$).

7. [2] Prove that the runtime of the code given below is Ω(logn)

```
(a). void read(int n) {
            int k = 1;
            while (k < n)
             k=k*3; }
```

Answer: The while loop will terminate once the value of 'k' is greater than or equal to the value of 'n'. The value of 'k' is multiplied by 3 in each iteration. If 'i' is the number of iterations, then 'k' has the value of 3i after i iterations. The loop is terminated upon reaching i iterations when $3^i \ge n \leftrightarrow i \ge \log_3 n$, which shows that i = Ω(logn)

8. [4] Let A be a two-dimensional array declared as follows:
A: array [1 …. 10] [1 …… 15] of integer;
Assuming that each integer takes one memory location, the array is stored in row-major order and the first element of the array is stored at location 100, what is the address of the element A[i][j]?

Answer: 15i + j + 84
The array is stored in row-major order, which means the elements are stored in the memory row-wise. When we are at the ith row, (i-1) rows with 15 elements in each row are already stored. To reach at jth element in the ith row, we are supposed to cross (j-1) elements.
Therefore, the address of the element A[i][j] = base address +(i-1)x15xM +(j-1)M

Where M is memory required to store one element. Here, it is given that each element of the array is integer of size 1 memory location and base address is 100.

address of the element A[i][j] = 100 + (i - 1) 15 + (j - 1)

= 15i + j + 84

9. [8] Suppose a polynomial in 3 variables (X, Y, Z) is stored via linked list representation, where each node stores the following fields:

COEF – coefficient, XEXP – power of X, YEXP - power of Y, ZEXP– power of Z, LINK – address of next node in the list

The following algorithm performs an insertion at the end of this linked list. Complete this algorithm.

INSPOLYEND (COEF, XEXP, YEXP, ZEXP, LINK, START, AVAIL, NC, NX, NY, NZ)

NC, NX, NY, NZ are the new terms corresponding to coefficient value and exponents for x, y, and z of the term. LINK, START and AVAIL have their usual meaning. LOC is a temporary pointer variable.

```
1.  [Overflow ?] If AVAIL = NULL Then write Overflow and exit
2.  [Remove 1st node from the AVAIL list]  Set NEW := Avail and AVAIL:= LINK[AVAIL]
3.  [Copy new data into new node]
        Set COEF[NEW]:=NC
            XEXP[NEW]:=NX
            YEXP[NEW]:=NY
            ZEXP[NEW]:= NZ
            LINK[NEW] := NULL
4.  [Is list empty ?]  If START = NULL then set START := NEW and Return
5.  [Initial search for last node]   LOC := START
6.  [Search for end of list]  Repeat while LINK[LOC] ≠ NULL
                                  LOC := LINK[LOC]
7.  [Set link field of last node to new node]  LINK[LOC] := NEW
8.  Return
```

10. [9] Fill in the blanks to complete the following function to insert in a sorted list:

```
struct node
{
    int data;
    struct node *next;
};
struct node  *start

void insert_sorted_list(int value)

{ /* insert a node with 'value' stored in the data field */
     struct node *New, *LocP,*Loc;
     New = (struct node *)malloc(sizeof (struct node));
     New  -> data = value;

     Loc = start;

     LocP = NULL;

     if(start = = NULL)

     {

          start = New;

         start -> next = NULL;

     }

     else

     {

         while (Loc -> data <  value)

          {

                LocP = Loc;

                Loc = Loc -> next;

          }

          New -> next = Loc;

          LocP -> next = New;

     }

}
```