National University of Computer and Emerging Sciences

Lahore Campus

Data Structures

Assignment #1

Date: 05-09-2025

Course code: CS2001

Instructor

Mr. Muhammad Naveed

Total marks: 120

Questions: 7

TA

Menahil Amir

_____ _____ _____

Roll No Section Student Signature

CLO 02: Evaluate different data structures in terms of memory complexity and time requirement

Attempt all the questions.

PART 1: TIME COMPLEXITY

Question 1: [Marks 10]

Perform a step count analysis on the Function given below and derive an equation for the worst case. Also compute the big-O.

```
void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

void func1(int arr[], int n)
{
  int i, j, min_idx;
  for (i = 0; i < n-1; i++)
  {
    min_idx = i;
    for (j = i+1; j < n; j++)
    if (arr[j] < arr[min_idx])
      min_idx = j;
      // Swap the found minimum element with the first element
    swap(&arr[min_idx], &arr[i]);
  } }
```

Question 2: [Marks 10]

Perform a step count analysis on the Function ***Pascal(int)*** and derive an equation for f(n) for the worst case.

```
void Pascal(int n){
long fcti, fctj, fcti_j;
for(int i=0;i<=n;i++){
for(int j=0;j<=i;j++){
fcti=fct(i);
fctj=fct(j);
fcti_j=fct(i-j);
cout<<fcti/(fctj*fcti_j)<<'\t';
}
cout<<endl;
}
}
```

```
long fct(int m){
if (m==0)
return 1;
long ans=1;
for(int i=1;i<=m;i++){
ans*=i;
}
return ans;
}
```

Question 3: [Marks 5+5+5+5+5=25]

Compute the time complexity of the following code snippets. Also find big-O

Note: You are required to show complete steps (with proper tables).

Problem # 1:

```
int i=1;
while(i<n)
{
cout<<i;
for(j=0;j<n;j=j+2)
{
cout<<j;
sum++;
}
cout<<sum;
i=i*4;
}
```

Problem # 2:

```
int i=1;
while(i<=n)
{
int j=1;
while(j<=i)
{
cout<<".";
j=j*2;
}
i=i*2;
}
```

Problem # 3:

```
sum=0;
for(i=0;j<=n;i++)
{

for(j=0;j<i*i;j++)
{
for(k=0;k<j;k++)
{
sum++;
}
}
}
```

Problem # 4:

```
for(int a=1,i=1; a<=n; i++)

{
a=a+i;
};
```

Problem # 5:

```
int sqrt(int N)
{
int d;
for(d=0; d*d<=N; d++)
{ }
return d-1;
}
bool primeNumber(int n)
{
bool isPrime = true;
for (int d=2; d<= sqrt(n) ;++d)
{
if (n%d==0)
return false;

}
return true;
}
```

Question 4: [Marks 5]

Write a (iterative method) binary search algorithm and then compute its time complexity.
Note: You are required to show complete steps.

CLO 01: Demonstrate basic concepts of data structure and algorithms.

PART 2: LINKED LIST

Question 1: [Marks 10]

A railway system maintains a **circular singly linked list** to represent the compartments of a train. Each node in the list contains:

- **Compartment Number** (unique identifier)
- **Passenger Count**

The train follows special operational rules at each station to optimize space and passenger distribution.

1. **Passenger Exit Rule:**
   When the train stops at a station, some passengers may leave each compartment (random or specified numbers).

2. **Minimum Passenger Rule:**
   After passenger exits, if a compartment has fewer than **3 passengers**, it must **merge with the next compartment**.
   - Merging means moving its passengers into the next compartment.
   - The current compartment is then **removed** from the circular linked list.

3. **Maximum Capacity Rule:**
   Each compartment can hold a maximum of **10 passengers**.
   - If merging causes a compartment to exceed capacity, the **extra passengers** are passed forward to the next compartment.
   - If overflow continues, passengers are redistributed circularly until everyone is seated.

4. **Circular Integrity:**
   The train always remains circular, i.e., the last compartment points back to the head.

Task

Simulate this process for a given train configuration (list of compartments with initial passenger counts) and a list of passengers leaving at a station. After applying the rules, output the final state of the train (compartment numbers and passenger counts).

Example:

Suppose a train has 5 compartments with initial passenger counts [5, 2, 8, 10, 4]. At Station 1, [1, 1, 3, 0, 2] passengers leave respectively. Show the final configuration of the train after applying the rules.

Question 2: [Marks 20]

Given a singly linked list and a positive integer k, write a function to rearrange the nodes in the list such that the nodes are grouped by k nodes into sub-lists. Within each sub-list, the nodes should be in their original relative order.

Example:

Input List: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9
k = 3
Output: [1 -> 2 -> 3] -> [4 -> 5 -> 6] -> [7 -> 8 -> 9]

If the list cannot be divided exactly on k, the last sub-list will be of the remaining nodes that are less than k.

Note: The output should be a linked list of linked lists, where each sub-list is represented as a separate linked list node.

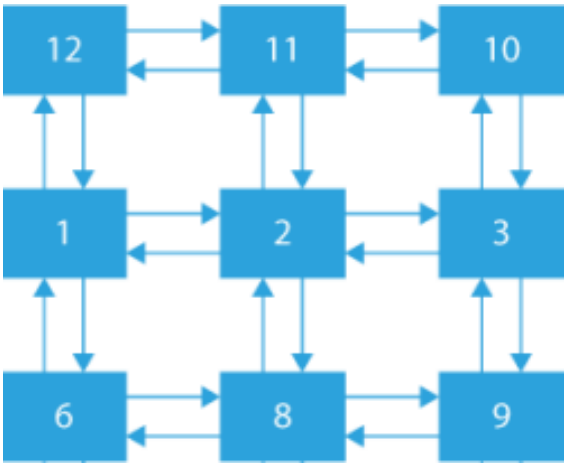
Question 3: [Marks 50]

Luck is the second most important thing to reach what you're looking for. Hard work and following the clues is the first. Your task is to reach the elite node in the network of linked lists.

Each node will have following members:

- *int data;*
- *node* up;*
- *node* down;*
- *node* left;*
- *node* right*

The entire structure will look like:



You have to make the maze using file, where each line represents a row with values seperated by commas. The file of the figure will look like :

12,11,10
1,2,3
6,8,9

You will start from the top left corner node and reach the next node using the data as the Clue.
Decode the clue as following:

- **Row of next node = (sum of all digits % No. of rows) +1**
- **Column of next node = number of digits**

Elite Node: The node whose clue will bring you to itself will be the elite node.

Functionalities:

- **Read(string filename)**

This method should be able to read data from a text file. Each line in the file contains all the entries in that row, values seperated by commas. There is one row per line.

12,34,56,78,90
44,76,34,87,99
88,65,12,19,50

This data shows that the maze will consist of 3 rows and 5 columns.

- **Print(node* head)**

Prints the maze created.

- **ClueRow(int data)**

Returns the row number of the next node to be visited (1st row is called row 1).

- **ClueColumn(int data)**

Returns the column number of the next node to be visited (1st column is called column 1)

- **Visited(node* Current)**

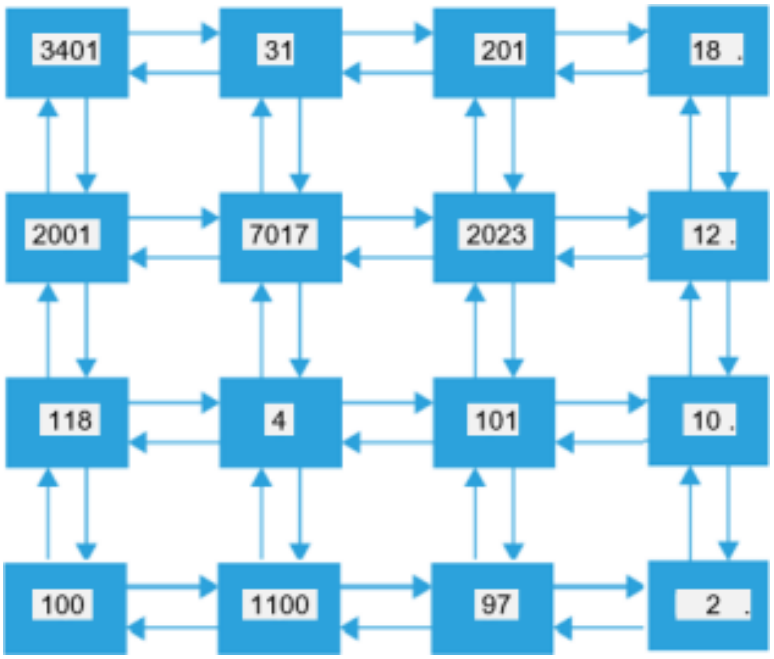
Prints the data of the node you're currently at. You need to print all the nodes you visit while reaching the elite node.

- **EliteNode(node* elite)**

When elite node is found, send it to this function to print the data and a message that elite node has been found. If the there doesn't exist any elite node, your program should be able to display a message for it. Also, if any node get's visited for the second time, the program should end after displaying a meaningful message.

***Example*:**

Input:



Output:

3401->18->7017->2->118->101
Elite Node = 101

YOU ARE NOT ALLOWED TO MAKE A 2D MATRIX AT ANY INSTANCE IN YOUR PROGRAM. READING DATA THROUGH FILE AFTER THE CREATION OF THE MAZE IS ALSO NOT ALLOWED.

Traverse the maze (left / right / top / bottom) to reach the next node.