**Multiple Choice Questions**

**Instructions**

1. **You cannot choose more than one**
2. **Negative marking**

1. The correct matching of the following pairs is

(a) Disk check              (1) Roundrobin

(b) Batch processing        (2) Scan

(c) Time sharing            (3) LIFO

(d) Stack operation         (4) FIFO

1. (a, 3) (b, 4) (c, 2) (d, 1)
2. (a, 4) (b, 3) (c, 2) (d, 1)
3. (a, 3) (b, 4) (c, 1) (d, 2)
4. **(a, 2) (b, 4) (c, 1) (d, 3)**

**Answer - D**

2. What does the following function do for a given Linked List with first node as head?

void fun1(struct node\* head)

{

  if(head == NULL)

    return;

  fun1(head->next);

  printf("%d  ", head->data);

}

1. Prints all nodes of linked lists
2. **Prints all nodes of linked list in reverse order**
3. Prints alternate nodes of Linked List
4. Prints alternate nodes in reverse order

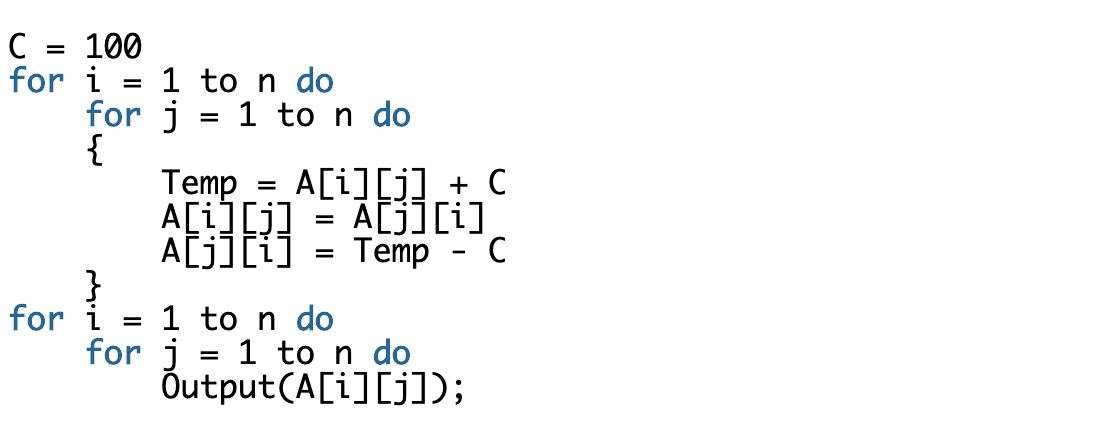
**Answer - B**

3. A program P reads in 500 integers in the range [0..100] representing the scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies?

1. **An array of 50 numbers**
2. An array of 100 numbers
3. An array of 500 numbers
4. A dynamically allocated array of 550 numbers

**Answer - A**

4. Let A be a square matrix of size n x n. Consider the following program. What is the expected output?



1. The matrix A itself
2. **Transpose of matrix A**
3. Adding 100 to the upper diagonal elements and subtracting 100 from diagonal elements of A
4. None of the above

**Answer - B**

5.    The \_\_\_\_\_\_\_\_\_ time in a swap out of a running process and swap in of a new process into the memory is very high.

**a.    context – switch**

b.    waiting

c.    execution

d.    all of the mentioned

**Answer - A**

6. Secure shell (SSH) network protocol is used for \_\_\_\_\_\_\_\_\_\_

a) secure data communication

b) remote command-line login

c) remote command execution

**d) all of the mentioned**

**Answer - D**

7. What is the time complexity of following code:

int i, j, k = 0;

for (i = n / 2; i <= n; i++) {

   for (j = 2; j <= n; j = j \* 2) {

       k = k + n / 2;

   }

}

1. O(n)
2. **O(nLogn)**
3. O(n^2)
4. O(n^2Logn)

**Answer - B**

8. Number of characters can fit in 2GB RAM? \_\_\_\_

**Answer - 2147483648 characters**

9. Which of the following are **not** shared by threads?

**A. program counter**

**B. stack**

C. Code section

D. Global variables

**Answer - A,B**

10. Which data structure is used in redo-undo feature?

1. **Stack**
2. Queue
3. Tree
4. Graph

**Answer - A**

11. What is recurrence for worst case of QuickSort and what is the time complexity in Worst case?

1. Recurrence is T(n) = T(n-2) + O(n) and time complexity is O(n^2)
2. **Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2)**
3. Recurrence is T(n) = 2T(n/2) + O(n) and time complexity is O(nLogn)
4. Recurrence is T(n) = T(n/10) + T(9n/10) + O(n) and time complexity is O(nLogn)

**Answer - B**

12. What is time complexity of fun()?

|  |
| --- |
| int fun(int n)  {    int count = 0;    for (int i = n; i > 0; i /= 2)       for (int j = 0; j < i; j++)          count += 1;    return count;  }   1. O(n^2) 2. **O(n\*Logn)** 3. O(n) 4. O(n\*Logn\*logn) |
| **Answer - B** |

13: An array consist of n elements.We want to create a min heap using the elements.The time complexity of building a heap will be in order of

1. O(logn)
2. **O(n)**
3. O(n \* logn)
4. O(n \* n)

**Answer - B**

14. Which of the following sorting algorithms has the lowest worst-case complexity?

**(A) Merge Sort**

(B) Bubble Sort

(C) Quick Sort

(D) Selection Sort

**Answer - A**

15. Which of the following transport layer protocols is used to support electronic mail?

**(A) SMTP**

(B) IP

(C) TCP

(D) UDP

**Answer - A**

16. A system has n resources R0,…,Rn-1,and k processes P0,….Pk-1.The implementation of the resource request logic of each process Pi is as follows:

if (i % 2 == 0) {

      if (i < n) request Ri

      if (i+2 < n) request Ri+2

}

else {

      if (i < n) request Rn-i

      if (i+2 < n) request Rn-i-2

}

In which one of the following situations is a deadlock possible?

(A) n=40, k=26

**(B) n=21, k=12**

(C) n=20, k=10

(D) n=41, k=19

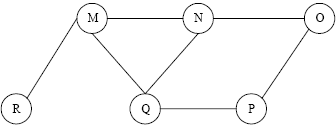
**Answer - B**

17. Which of the following data structure/s is best suited for converting **recursive implementation to iterative implementation** of an algorithm?

1. Queue
2. **Stack**
3. Tree
4. Graph

**Answer - D**

18. The **Breadth First Search** algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is



1. MNOPQR
2. NQMPOR
3. **QMNPRO**
4. QMNPOR

**Answer - C**

**Programming Questions**

**Instructions**:

1. Write code in programming language of your choice.
2. With each question also **mention time and space complexity**.
3. While evaluating these questions, the main emphasis will be on correctness of main algorithm
4. You can use standard library functions like sort, min, max.

**1)** Suppose you have a deck of cards represented as a linked list. You can perfectly shuffle

that list by cutting it at the halfway point, then interleaving the two halves by alternating back and forth between the cards. For example, suppose you want to perfectly shuffle

this sequence:

 1 2 3 4 5 6 7 8 9 10

You'd start by splitting it into two halves, like this:

 1 2 3 4 5 6 7 8 9 10

Then, you'd interleave the halves, like this:

6 1 7 2 8 3 9 4 10 5

The resulting list is said to have been perfectly shuffled. Your job is to write a function that accepts as input a linked list with an even number of elements, then rearranges the elements in that list so that they're perfectly shuffled.

**Solution -**

(C)

**Function to be called is Node\* shuffle()**

Time complexity - O(N)

Space complexity - O(1)

void swap(Node\* head1, Node\* head2) {

Node\* tail = NULL;

while (head2 != NULL) {

if (tail == NULL) {

tail = head2;

}

else {

tail->next = head2;

tail = head2;

}

Node\* next = head2->next;

head2->next = NULL;

head2 = next;

// swapping the lists

Node\* temp = head1;

head1 = second;

second = head1;

}

}

# if the **head node** is provided as the input for the list

Node\* shuffle(Node\* head) {

Node\* fast = first;

while (fast->next->next != NULL) {

fast = fast->next->next;

first = first->next;

}

/\* Split the list at this point. \*/

Node\* head2 = first->next; // head2 points to the start of the 2nd list

first->next = NULL;

// call swap function to shuffle the 2 lists

swap(head,head2);

return head; // head will be the head node of the shuffled list

}

**2)** Given a number n, generate all distinct ways to write n as the sum of positive integers.

For example, with n = 4, the options are 4, 3 + 1, 2 + 2, 2 + 1 + 1, and 1 + 1 + 1 + 1.

**Solution -**

(Python)

Time complexity - O(2^N)

Space complexity - O(N^2)

n = int(input())

result = [] # this array will contain all the distinct ways

def generate(n,ind,temparr):

if (n == 0):

if (len(temparr) != 1):

result.append(temparr) # append each distinct way to the result array

for j in range(ind, n + 1):

temparr.append(j)

generate(n - j,j)

temparr = temparr[:-1]

temparr = []

ind = 1

generate(n,ind,temparr)

# display output

for i in result:

print(i)

**3)** Given a list of x, y values that represent points in an x, y coordinate system. Process the list of x, y coordinates to determine which two are the closest.

[C++]

Time complexity - O(NlogN)

Space complexity - O(N)

#include <iostream>

#include<cmath>

#include<algorithm>

using namespace std;

struct point {

int x, y;

};

int cmpX(point p1, point p2) { //to sort according to x value

return (p1.x < p2.x);

}

int cmpY(point p1, point p2) { //to sort according to y value

return (p1.y < p2.y);

}

float dist(point p1, point p2) { //find distance between p1 and p2

return sqrt((p1.x - p2.x)\*(p1.x - p2.x) + (p1.y - p2.y)\*(p1.y - p2.y));

}

float findMinDist(point pts[], int n) { //find minimum distance between two points in a set

float min = 9999;

for (int i = 0; i < n; ++i)

for (int j = i+1; j < n; ++j)

if (dist(pts[i], pts[j]) < min)

min = dist(pts[i], pts[j]);

return min;

}

float min(float a, float b) {

return (a < b)? a : b;

}

float stripClose(point strip[], int size, float d) { //find closest distance of two points in a strip

float min = d;

for (int i = 0; i < size; ++i)

for (int j = i+1; j < size && (strip[j].y - strip[i].y) < min; ++j)

if (dist(strip[i],strip[j]) < min)

min = dist(strip[i], strip[j]);

return min;

}

float findClosest(point xSorted[], point ySorted[], int n){

if (n <= 3)

return findMinDist(xSorted, n);

int mid = n/2;

point midPoint = xSorted[mid];

point ySortedLeft[mid+1]; // y sorted points in the left side

point ySortedRight[n-mid-1]; // y sorted points in the right side

intleftIndex = 0, rightIndex = 0;

for (int i = 0; i < n; i++) { //separate y sorted points to left and right

if (ySorted[i].x <= midPoint.x)

ySortedLeft[leftIndex++] = ySorted[i];

else

ySortedRight[rightIndex++] = ySorted[i];

}

float leftDist = findClosest(xSorted, ySortedLeft, mid);

float rightDist = findClosest(ySorted + mid, ySortedRight, n-mid);

float dist = min(leftDist, rightDist);

point strip[n]; //hold points closer to the vertical line

int j = 0;

for (int i = 0; i < n; i++)

if (abs(ySorted[i].x - midPoint.x) <dist) {

strip[j] = ySorted[i];

j++;

}

return min(dist, stripClose(strip, j, dist)); //find minimum using dist and closest pair in strip

}

float minDist(point pts[], int n) { //find distance of closest pair in a set of points

point xSorted[n];

point ySorted[n];

for (int i = 0; i < n; i++) {

xSorted[i] = pts[i];

ySorted[i] = pts[i];

}

sort(xSorted, xSorted+n, cmpX);

sort(ySorted, ySorted+n, cmpY);

return findClosest(xSorted, ySorted, n);

}

int main() {

point array[] ={{1, 2}, {10, 20}, {2, 3}, {4, 7}};

int size = 4;

cout<< minDist(array, size);

}