UNBXD

Multiple Choice Questions

Instructions

a. You cannot choose more than one
b. Negative marking

Note: All the answers are marked in BOLD

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

A. (a, 3) (b, 4) (c, 2) (d, 1)
B. (a, 4) (b, 3) (c, 2) (d, 1)
C. (a, 3) (b, 4) (c, 1) (d, 2)
D. (a, 2) (b, 4) (c, 1) (d, 3)

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);
}
```

A. Prints all nodes of linked lists
B. Prints all nodes of linked list in reverse order
C. Prints alternate nodes of Linked List
D. Prints alternate nodes in reverse order

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?

A. An array of 50 numbers
B. An array of 100 numbers
C. An array of 500 numbers
D. A dynamically allocated array of 550 numbers

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

UNBXD

```
C = 100
for i = 1 to n do
    for j = 1 to n do
    {
        Temp = A[i][j] + C
        A[i][j] = A[j][i]
        A[j][i] = Temp - C
    }
for i = 1 to n do
    for j = 1 to n do
        Output(A[i][j]);
```

A. The matrix A itself
B. Transpose of matrix A
C. Adding 100 to the upper diagonal elements and subtracting 100 from diagonal elements of A
D. None of the above

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**

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**

7. What is the time complexity of following code:

```
int i, j, k = 0;

for (i = n / 2; i <=
  for (j = 2; j <= n; j = j * 2) {
```

```
        k = k + n / 2;

      }
    }
```

a. O(n)
b. O(nLogn)
c. O(n^2)
d. O(n^2Logn)

8. Number of characters can fit in 2GB RAM? **ans: 2000000000 chars**

9. Which of the following are **not** shared by threads?
A. program counter
B. stack
C. **Code section**
D. Global variables

10. Which data structure is used in redo-undo feature?
A. Stack
B. Queue
C. Tree
D. Graph

11. What is recurrence for worst case of QuickSort and what is the time complexity in Worst case?
A. Recurrence is T(n) = T(n-2) + O(n) and time complexity is O(n^2)
B. Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2)
C. Recurrence is T(n) = 2T(n/2) + O(n) and time complexity is O(nLogn)
D. Recurrence is T(n) = T(n/10) + T(9n/10) + O(n) and time complexity is O(nLogn)

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;
}
```

A. O(n^2)
B. O(n*Logn)
C. O(n)
D. O(n*Logn*logn)

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

a. O(logn)
b. O(n)
c. O(n * logn)
d. O(n * n)

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

15. Which of the following transport layer protocols is used to support electronic mail?
(**A) SMTP**
(B) IP
(C) TCP
(D) UDP

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
}
```

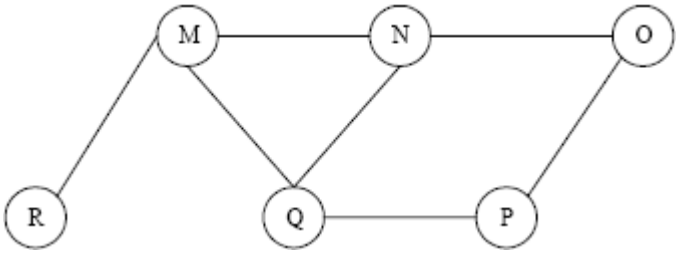
In which one of the following situations is a deadlock possible?
(A) n=40, k=26
(B) n=21, k=12
(C) n=20, k=10
(D) n=41, k=19

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

A. Queue
B. Stack
C. Tree

D. Graph

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



A. MNOPQR
B. NQMPOR
C. QMNPRO
D. QMNPOR

UNBXD

Programming Questions

Instructions:

a. Write code in programming language of your choice.
b. With each question also **mention time and space complexity**.
c. While evaluating these questions, the main emphasis will be on correctness of main algorithm
d. 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.

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.

code: the code is in c++

```
#include <bits/stdc++.h>
using namespace std;

int countWays(int n)
{

   int table[n+1];

   memset(table, 0, sizeof(table));


   table[0] = 1;

   for (int i=1; i<n; i++)
      for (int j=i; j<=n; j++)
         table[j] += table[j-i];
```

```
    return table[n];
}

int main()
{
    int n = 10;
    cout << countWays(n);
    return 0;
}
```

the time complexity is O(n^2)
the space complexity is O(n)

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.
code: code is in c++

```
#include <bits/stdc++.h>
using namespace std;


class Point
{
    public:
    int x, y;
};
int compareX(const void* a, const void* b)
{
    Point *p1 = (Point *)a, *p2 = (Point *)b;
    return (p1->x - p2->x);
}


int compareY(const void* a, const void* b)
{
    Point *p1 = (Point *)a, *p2 = (Point *)b;
    return (p1->y - p2->y);
}
float dist(Point p1, Point p2)
{
    return sqrt( (p1.x - p2.x)*(p1.x - p2.x) +
                 (p1.y - p2.y)*(p1.y - p2.y)
               );
}

float bruteForce(Point P[], int n)
{
    float min = FLT_MAX;
```

```
   for (int i = 0; i < n; ++i)
      for (int j = i+1; j < n; ++j)
         if (dist(P[i], P[j]) < min)
            min = dist(P[i], P[j]);
   return min;
}


float min(float x, float y)
{
   return (x < y)? x : y;
}



float stripClosest(Point strip[], int size, float d)
{
   float min = d;

   qsort(strip, size, sizeof(Point), compareY);

   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 closestUtil(Point P[], int n)
{

   if (n <= 3)
      return bruteForce(P, n);

   int mid = n/2;
   Point midPoint = P[mid];


   float dl = closestUtil(P, mid);
   float dr = closestUtil(P + mid, n - mid);


   float d = min(dl, dr);


   Point strip[n];
   int j = 0;
```

```
   for (int i = 0; i < n; i++)
      if (abs(P[i].x - midPoint.x) < d)
         strip[j] = P[i], j++;


   return min(d, stripClosest(strip, j, d) );
}


float closest(Point P[], int n)
{
   qsort(P, n, sizeof(Point), compareX);

   return closestUtil(P, n);
}


int main()
{
   Point P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};
   int n = sizeof(P) / sizeof(P[0]);
   cout << "The smallest distance is " << closest(P, n);
   return 0;
}
```

the time complexity is T(n) = 2T(n/2) + O(n) + O(nLogn) + O(n)