```
#include<stdio.h>
#include<sys/types.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<ctype.h>



int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {
      // If current element is smaller than the pivot
      if (a[j] < pivot)
      {
         i++; // increment index of smaller element
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
```

```
/* function to implement quick sort */
void quick(int a[], int start, int end) /* a[] = array to be sorted, start = Starting index, end = Ending index */
{
   if (start < end)
   {
      int p = partition(a, start, end); //p is the partitioning index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}

void merge(int arr[], int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
```

```
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
```

```
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
  pid_t p;
  int n;
  printf("Enter the number of elements");
   scanf("%d",&n);
   int a[n];

   for(int i=0;i<n;i++)
  {
    printf("Enter %d th element  ",(i+1));
    scanf("%d",&a[i]);
   }

   p=fork();
```

```
 quick(a,0,n-1);
  if(p==0)
   {

     printf("Process is child, ID is %d \n",getpid());
     printf(" Parent's process, ID is %d \n",getppid());
     quick(a,0,n-1);
    printf("After sorting elemets are ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }

  }

  else{
      printf("Process is in Parent ,ID is %d \n",getpid());
     mergeSort(a,0,n-1);
     printf("After merge Sort elements are \n ");

    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }
    }
   return 0;
}
```

Plagiarism Detector .net

Nov 08, 2022

Plagiarism Scan Report

Characters:6962	Words:996
Sentences:66	Speak Time: 8 Min

Excluded URL	None

Content Checked for Plagiarism

Blockchain technology is a distributed ledger with data entries that are disseminated among network nodes and contain all the specifics of completed transactions. Consensus procedures certify each transaction done in the system, and the data that is saved cannot be changed. The key technology underlying the widely used cryptocurrency, Bitcoin is called

Total No. of Questions : 8]

SEAT No. :

P820

[Total No. of Pages : 2

[5870] - 1141
T.E. (I.T.)
OPERATING SYSTEMS
(2019 Pattern) (Semeser - I) (314442)

Time : 2½ Hours] *[Max. Marks : 70*

Instructions to the candidates:

1) ***Answer Q1or Q2, Q3 or Q4, Q5 or Q6, Q7 or Q8.***
2) ***Neat diagram must be drawn wherever necessary.***
3) ***Figures to the right indicate full marks.***
4) ***Assume suitable data, if necessary.***

Q1) a) Explain requirements for mutual exclusion. **[8]**

b) Explain following terms: **[9]**

i) Principles of concurrency

ii) Semaphores

iii) Mutex

OR

Q2) a) Explain following synchronization problems **[8]**

i) Readers/ Writers Problem.

ii) Producer and Consumer problem.

b) Explain Inter-process communication using **[9]**

i) Pipes

ii) Shared Memory

iii) Semaphore

Q3) a) For the given reference string with 4 page frame abvailable, determine the number of page faults for FIFO, OPTIMAL, LRU algorithms:
3, 5, 3, 7, 2, 1, 5, 4, 6, 7, 4, 1, 2. **[12]**

b) Explain with the help of neat diagram Hierarchical page table. **[6]**

OR

Q4) a) A1 MByte block of memory is allocated using the buddy system. **[10]**

i) Show the result of the following sequence in the graphical form for A : Request 35KB, B: Request 140 KB, C : Request 120KB, D : Request 250KB, Return C, E: Request 70 KB, Return B, Return D, F : Request, 100KB, Return A, Return E.

ii) Draw the tree representation after Return B.

b) What is Principle of Locality? Explain working set model. **[8]**

P.T.O.

Q5) a) Assume a disk with 200 tracks and the disk request queue has random Requests in it as follows: 55, 58, 39, 18, 90, 160, 150, 38, 184. Find the no of tracks traversed and average seek length if **[12]**

i) FCFS

ii) SSTF

iii) SCAN

iv) C-Look is used and initially head is at track no 100. Assume head is moving towards outer track for SCAN and C-Look.

b) State and explain diff approaches of I/O buffering. **[6]**

OR

Q6) a) Describe 3 methods of record blocking with the help of neat diagrams. **[9]**

b) Explain different file organization techniques. **[9]**

Q7) a) Explain working of "General Loading Scheme" with advantages and disadvantages. **[6]**

b) Draw and explain flowchart of Pass-2 of two pass assembler. **[6]**

c) Define following system software components with suitable diagram. **[5]**

i) Compiler

ii) Loader

iii) Editor

iv) Linker

v) Debugger

OR

Q8) a) Explain Phase structure of Compiler with neat diagram. **[6]**

b) What is the need of symbol table (ST) and literal table (LT) in two pass assembler? Explain fields of ST and LT with suitable example. **[6]**

c) What are types of loaders? Discuss four different functions of loaders. **[5]**

♦ ♦ ♦

Total No. of Questions : 8]

SEAT No. : [ ]

P806

[Total No. of Pages : 2

[5870]-1126
T.E. (Computer Engineering)
THEORY OF COMPUTATIONS
(2019 Pattern) (Semester-I) (310242)

Time : 2½ Hours] ***[Max. Marks : 70***

Instructions to the candidates:

1) ***Answer Q1 or Q2, Q3 or Q4, Q5 or Q6, Q7 or Q8.***
2) ***Figures to the right side indicate full marks.***
3) ***Neat diagrams must be drawn wherever necessary.***
4) ***Assume suitable data, if necessary.***

Q1) a) Write a grammar G for generating the language **[9]**
i) L={w belongs to {a,b}* | w is an even length palindrome with |w|>0}
ii) Set of odd length strings in {0,1}* with middle symbol '1'

b) Simplify the following grammar **[9]**
S→ 0A0|1B1|BB
A→ C
B→ S|A
C→ S|€

OR

Q2) a) Reduce the following grammar to Greibach Normal form. **[9]**
S→ AA | 0
A→ SS | 1

b) Construct a DFA for the following left linear grammar. **[9]**
S→ B1/A0/C0
B→ B1/1
A→ A1/B1/C0
C→ A0

Q3) a) Construct a context free grammar which accepts N(A), where **[9]**
A = ({q0,q1}, {0,1}, {Z0,Z}, δ, q0, Z0, φ}-where δ is given by
δ (q0, 1, Z0) = {(q0, ZZ0)}
δ (q0, ε, Z0) = {(q0,ε)}
δ (q0, 1, Z) = {(q0, Z Z)}
δ (q0, 0, Z) = {(q1, Z)}
δ (q1, 1, Z) = {(q1, ε)}
δ (q1, 0, Z0) = {(q0, Z0)}

P.T.O.

b) Construct a PDA that accept the language generated by grammar **[8]**

i) $S \rightarrow 0S1|A, A \rightarrow 1A0|S|€$

ii) $S \rightarrow aABB|aAA, A \rightarrow aBB|a, B \rightarrow bAA|A$

OR

Q4) a) What is NPDA? Construct a NPDA for the set of all strings over {a,b} with odd length palindrome. **[9]**

b) Design a push down automaton to recognize the language generated by the following grammar: **[8]**

$S \rightarrow S + S \mid S \square S \mid 4 \mid 2$

Show the acceptance of the input string 2 + 2*4 by this PDA.

Q5) a) What is a Turing Machine? Give the formal definition of TM. **[9]**
Design a TM that replaces every occurrence of abb by baa.

b) What are the different ways for extension of TM? Explain. **[9]**
Design TM for language L = {a^i b^j | i<j}

OR

Q6) a) What is TM? Design TM to check well formedness of Parenthesis. Expand the transition for (())() **[9]**

b) Elaborate the following terms **[9]**

i) Universal Turing Machine (UTM)

ii) Recursively Enumerable Languages

iii) Halting Problem of Turing Machine

Q7) a) Justify "Halting Problem of Turing machine is undecidable". **[9]**

b) Define the Class P and Class NP and Problem with their example in detail. **[8]**

OR

Q8) a) Explain Satisfiability Problem and SAT Problem and comment on NP Completeness of the SAT Problem. **[9]**

b) What do you mean by polynomial time reduction? Explain with suitable example. **[8]**

Total No. of Questions : 8]

SEAT No. :

P821

[Total No. of Pages : 2

[5870]-1142
T.E. (Information Technology)
MACHINE LEARNING
(2019 Pattern) (Semester - I) (314443)

Time : 2½ Hours] ***[Max. Marks : 70***

Instructions to the candidates:

1) ***Answers Q1 or Q2, Q3 or Q4, Q5 or Q6, Q7 or Q8.***
2) ***Neat diagrams must be drawn wherever necessary.***
3) ***Figures to the right side indicate full marks.***
4) ***Assume suitable data if necessary.***

Q1) a) Why do we need optimization in Regression? Explain gradient descent optimization technique in detail. **[9]**

b) What do you meant by least square method? Explain least square method in the context of linear regression. **[8]**

OR

Q2) a) What is overfitting and underfitting? Explain the reasons of overfitting and underfitting. **[9]**

b) What is multivariate and univariate regression state the difference with examples. **[8]**

Q3) a) Explain ID-3 Decision tree algorithm in detail. **[9]**

b) Explain the measures of impurity (Information Gain, Gini Index, Entropy) **[6]**

c) What are the advantages and limitation of tree-based model. **[2]**

OR

Q4) a) Explain in brief the Bayesian network for learning and inference. **[9]**

b) Explain naive bayes algorithm with example. **[6]**

c) Enlist any four applications of naive bayes classifier. **[2]**

P.T.O.

Q5) a) Following is a dataset for weight of teens and whether they like Pizza. **[6]**

Weight	Like Pizza?
78	YES
54	NO
69	YES
73	YES
59	NO
48	NO
82	NO
65	YES

Using K-Nearest Neighbours (kNN) classification algorithm determine if a teen weighing 63 kgs likely to like pizza? (Use K=3)

b) Explain Agglomerative hierarchical clustering. **[6]**

c) Explain association rule mining. Explain the various Approaches to improve the efficiency of Apriori algorithm. **[6]**

OR

Q6) a) Explain the any two algorithms used for clustering. **[6]**

b) Explain following terms: **[6]**

a) Centroid
b) Medoid
c) Dendrogram

c) Explain association rule mining. Comment on role of support and confidence in associations of rule mining? **[6]**

Q7) a) Explain what is Deep Learning and its different architectures? State the various applications of deep learning? **[9]**

b) Explain how the learning parameters are updated for Multilayer Perceptron? **[9]**

OR

Q8) a) Explain the following activation functions. **[9]**

i) Sigmoid ii) Tanh
iii) ReLU

b) What is the difference between biological neuron and artificial neuron? Explain the simulation of AND gate using McCulloch Pitts Neuron? **[9]**

□□□

Total No. of Questions : 8]

SEAT No. : ____

P2326

[Total No. of Pages : 2

[5870]-1144
T.E. (Information Technology)
DESIGN & ANALYSIS OF ALGORITHMS
(2019 Pattern) (Semester - I) (314445A) (Elective - I)

Time : 2½ Hours] ***[Max. Marks : 70***

Instructions to the candidates:

1) ***Answer Q1 or Q2, Q3 or Q4, Q5 or Q6, Q7or Q8.***
2) ***Neat diagrams must be drawn wherever necessary.***
3) ***Figures to the right side indicate full marks.***
4) ***Assume suitable data, if necessary.***

Q1) a) Consider 0/1 knapsack problem N = 3:W=(4,6,8) and P=(10, 12, 15).by using dynamic programming determine the optimal profit for knapsack capacity 10? **[9]**

b) Explain coin change Making problem in detail? **[9]**

OR

Q2) a) Explain how dynamic programming is used to obtain optimal solution for travelling salesperson problem. also explain why this technique is not used to solve TSP for large number of cities? **[9]**

b) What is dynamic programming? Is this the optimization technique? Give reasons what are its drawbacks? **[9]**

Q3) a) Find all possible solutions for 5 queens problem using backtracking. **[9]**

b) Current configuration is (7,5,3,l) for 8 queens problem. Find the answer tuplc using backtracking method. **[8]**

OR

Q4) a) State the principle of backtracking. Explain the constraints used in backtracking with an example. **[9]**

b) What is m colorability optimization problem. Explain with an example. **[8]**

Q5) a) Differentiate between backtracking & branch and bound. Illustrate with example of Knapsack problem. **[9]**

P.T.O.

b) Solve following Job sequencing with deadline problem using Branch and Bound. **[9]**

Job	P	d	t
1	5	1	1
2	10	3	2
3	6	2	1
4	3	1	1

OR

Q6) a) Solve the following instance of the knapsack problem by branch and bound algorithm for W=16. **[9]**

Item	Weight	Value in Rs.
1	10	100
2	7	63
3	8	56
4	4	12

b) Describe the following with respect to B & B **[9]**

- The method
- LC search
- Control abstraction for LC search
- Bounding function

Q7) a) When do you claim that algorithm is polynomial time algorithm? Explain with an example. **[9]**

b) Explain i) Complexity Classes ii) Deterministic Algorithms. **[8]**

OR

Q8) a) Explain Vertex cover problem is in detail. **[9]**

b) What is deterministic algorithm? Write any one deterministic algorithm. **[8]**

❄ ❄ ❄

```
#include<stdio.h>
#include<sys/types.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<ctype.h>



int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {
      // If current element is smaller than the pivot
      if (a[j] < pivot)
      {
         i++; // increment index of smaller element
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
```

```
/* function to implement quick sort */
void quick(int a[], int start, int end) /* a[] = array to be sorted, start = Starting index, end = Ending index */
{
   if (start < end)
   {
      int p = partition(a, start, end); //p is the partitioning index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}

void merge(int arr[], int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
```

```
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
```

```
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
  pid_t p;
  int n;
  printf("Enter the number of elements");
   scanf("%d",&n);
   int a[n];

   for(int i=0;i<n;i++)
  {
    printf("Enter %d th element  ",(i+1));
    scanf("%d",&a[i]);
   }

   p=fork();
```

```
 quick(a,0,n-1);
  if(p==0)
   {

     printf("Process is child, ID is %d \n",getpid());
     printf(" Parent's process, ID is %d \n",getppid());
     quick(a,0,n-1);
    printf("After sorting elemets are ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }

  }

  else{
      printf("Process is in Parent ,ID is %d \n",getpid());
     mergeSort(a,0,n-1);
     printf("After merge Sort elements are \n ");

    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }
    }
   return 0;
}
```

Plagiarism Detector .net

Nov 08, 2022

Plagiarism Scan Report

Characters:6962	Words:996
Sentences:66	Speak Time: 8 Min

Excluded URL	None

Content Checked for Plagiarism

Blockchain technology is a distributed ledger with data entries that are disseminated among network nodes and contain all the specifics of completed transactions. Consensus procedures certify each transaction done in the system, and the data that is saved cannot be changed. The key technology underlying the widely used cryptocurrency, Bitcoin is called

```
#include<stdio.h>
#include<sys/types.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<ctype.h>



int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {
      // If current element is smaller than the pivot
      if (a[j] < pivot)
      {
         i++; // increment index of smaller element
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
```

```
/* function to implement quick sort */
void quick(int a[], int start, int end) /* a[] = array to be sorted, start = Starting index, end = Ending index */
{
   if (start < end)
   {
      int p = partition(a, start, end); //p is the partitioning index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}

void merge(int arr[], int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
```

```
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
```

```
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
  pid_t p;
  int n;
  printf("Enter the number of elements");
   scanf("%d",&n);
   int a[n];

   for(int i=0;i<n;i++)
  {
    printf("Enter %d th element  ",(i+1));
    scanf("%d",&a[i]);
   }

   p=fork();
```

```
 quick(a,0,n-1);
  if(p==0)
   {

     printf("Process is child, ID is %d \n",getpid());
     printf(" Parent's process, ID is %d \n",getppid());
     quick(a,0,n-1);
    printf("After sorting elemets are ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }

  }

  else{
      printf("Process is in Parent ,ID is %d \n",getpid());
     mergeSort(a,0,n-1);
     printf("After merge Sort elements are \n ");

    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }
    }
   return 0;
}
```

Plagiarism Detector .net

Nov 08, 2022

Plagiarism Scan Report

Characters:6962	Words:996
Sentences:66	Speak Time: 8 Min

Excluded URL	None

Content Checked for Plagiarism

Blockchain technology is a distributed ledger with data entries that are disseminated among network nodes and contain all the specifics of completed transactions. Consensus procedures certify each transaction done in the system, and the data that is saved cannot be changed. The key technology underlying the widely used cryptocurrency, Bitcoin is called

```
#include<stdio.h>
#include<sys/types.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<ctype.h>



int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {
      // If current element is smaller than the pivot
      if (a[j] < pivot)
      {
         i++; // increment index of smaller element
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
```

```
/* function to implement quick sort */
void quick(int a[], int start, int end) /* a[] = array to be sorted, start = Starting index, end = Ending index */
{
   if (start < end)
   {
      int p = partition(a, start, end); //p is the partitioning index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}

void merge(int arr[], int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
```

```
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
```

```
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
  pid_t p;
  int n;
  printf("Enter the number of elements");
   scanf("%d",&n);
   int a[n];

   for(int i=0;i<n;i++)
  {
    printf("Enter %d th element  ",(i+1));
    scanf("%d",&a[i]);
   }

   p=fork();
```

```
 quick(a,0,n-1);
  if(p==0)
   {

     printf("Process is child, ID is %d \n",getpid());
     printf(" Parent's process, ID is %d \n",getppid());
     quick(a,0,n-1);
    printf("After sorting elemets are ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }

  }

  else{
      printf("Process is in Parent ,ID is %d \n",getpid());
     mergeSort(a,0,n-1);
     printf("After merge Sort elements are \n ");

    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }
    }
   return 0;
}
```

Plagiarism Detector .net

Nov 08, 2022

Plagiarism Scan Report

Characters:6962	Words:996
Sentences:66	Speak Time: 8 Min

Excluded URL	None

Content Checked for Plagiarism

Blockchain technology is a distributed ledger with data entries that are disseminated among network nodes and contain all the specifics of completed transactions. Consensus procedures certify each transaction done in the system, and the data that is saved cannot be changed. The key technology underlying the widely used cryptocurrency, Bitcoin is called

```
#include<stdio.h>
#include<sys/types.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<ctype.h>



int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {
      // If current element is smaller than the pivot
      if (a[j] < pivot)
      {
         i++; // increment index of smaller element
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
```

```
/* function to implement quick sort */
void quick(int a[], int start, int end) /* a[] = array to be sorted, start = Starting index, end = Ending index */
{
   if (start < end)
   {
      int p = partition(a, start, end); //p is the partitioning index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}

void merge(int arr[], int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
```

```
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
```

```
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
  pid_t p;
  int n;
  printf("Enter the number of elements");
   scanf("%d",&n);
   int a[n];

   for(int i=0;i<n;i++)
  {
    printf("Enter %d th element  ",(i+1));
    scanf("%d",&a[i]);
   }

   p=fork();
```

```
 quick(a,0,n-1);
  if(p==0)
   {

     printf("Process is child, ID is %d \n",getpid());
     printf(" Parent's process, ID is %d \n",getppid());
     quick(a,0,n-1);
    printf("After sorting elemets are ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }

  }

  else{
      printf("Process is in Parent ,ID is %d \n",getpid());
     mergeSort(a,0,n-1);
     printf("After merge Sort elements are \n ");

    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }
    }
   return 0;
}
```

Plagiarism Detector .net

Nov 08, 2022

Plagiarism Scan Report

Characters:6962	Words:996
Sentences:66	Speak Time: 8 Min

Excluded URL	None

Content Checked for Plagiarism

Blockchain technology is a distributed ledger with data entries that are disseminated among network nodes and contain all the specifics of completed transactions. Consensus procedures certify each transaction done in the system, and the data that is saved cannot be changed. The key technology underlying the widely used cryptocurrency, Bitcoin is called

```
#include<stdio.h>
#include<sys/types.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<ctype.h>



int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {
      // If current element is smaller than the pivot
      if (a[j] < pivot)
      {
         i++; // increment index of smaller element
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
```

```
/* function to implement quick sort */
void quick(int a[], int start, int end) /* a[] = array to be sorted, start = Starting index, end = Ending index */
{
   if (start < end)
   {
      int p = partition(a, start, end); //p is the partitioning index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}

void merge(int arr[], int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
```

```
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
```

```
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
  pid_t p;
  int n;
  printf("Enter the number of elements");
   scanf("%d",&n);
   int a[n];

   for(int i=0;i<n;i++)
  {
    printf("Enter %d th element  ",(i+1));
    scanf("%d",&a[i]);
   }

   p=fork();
```

```
 quick(a,0,n-1);
  if(p==0)
   {

     printf("Process is child, ID is %d \n",getpid());
     printf(" Parent's process, ID is %d \n",getppid());
     quick(a,0,n-1);
    printf("After sorting elemets are ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }

  }

  else{
      printf("Process is in Parent ,ID is %d \n",getpid());
     mergeSort(a,0,n-1);
     printf("After merge Sort elements are \n ");

    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }
    }
   return 0;
}
```

Plagiarism Detector .net

Nov 08, 2022

Plagiarism Scan Report

Characters:6962	Words:996
Sentences:66	Speak Time: 8 Min

Excluded URL	None

Content Checked for Plagiarism

Blockchain technology is a distributed ledger with data entries that are disseminated among network nodes and contain all the specifics of completed transactions. Consensus procedures certify each transaction done in the system, and the data that is saved cannot be changed. The key technology underlying the widely used cryptocurrency, Bitcoin is called

```
#include<stdio.h>
#include<sys/types.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<ctype.h>



int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {
      // If current element is smaller than the pivot
      if (a[j] < pivot)
      {
         i++; // increment index of smaller element
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
```

```
/* function to implement quick sort */
void quick(int a[], int start, int end) /* a[] = array to be sorted, start = Starting index, end = Ending index */
{
   if (start < end)
   {
      int p = partition(a, start, end); //p is the partitioning index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}

void merge(int arr[], int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
```

```
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
```

```
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

int main()
{
  pid_t p;
  int n;
  printf("Enter the number of elements");
   scanf("%d",&n);
   int a[n];

   for(int i=0;i<n;i++)
  {
    printf("Enter %d th element  ",(i+1));
    scanf("%d",&a[i]);
   }

   p=fork();
```

```
 quick(a,0,n-1);
  if(p==0)
   {

     printf("Process is child, ID is %d \n",getpid());
     printf(" Parent's process, ID is %d \n",getppid());
     quick(a,0,n-1);
    printf("After sorting elemets are ");
    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }

  }

  else{
      printf("Process is in Parent ,ID is %d \n",getpid());
     mergeSort(a,0,n-1);
     printf("After merge Sort elements are \n ");

    for(int i=0;i<n;i++)
    {
      printf("%d ",a[i]);
    }
    }
   return 0;
}
```

Plagiarism Detector .net

Nov 08, 2022

Plagiarism Scan Report

Characters:6962	Words:996
Sentences:66	Speak Time: 8 Min

Excluded URL	None

Content Checked for Plagiarism

Blockchain technology is a distributed ledger with data entries that are disseminated among network nodes and contain all the specifics of completed transactions. Consensus procedures certify each transaction done in the system, and the data that is saved cannot be changed. The key technology underlying the widely used cryptocurrency, Bitcoin is called