VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAALAB

Name K.S.I.SIVANI Roll No. 1602-21-733-052 Page No. 1

PRELAB QUESTIONS-1:

1) Define an algorithm:

A: A finite set of instructions for solving a particular task.

2) List the criteria to design an algorithm.

A: Input, Output, Finiteness, Definiteness, Effectiveness.

3) Write the pseudocode conventions for conditional statements, control loops and record data type.

* → Looping statements:

→ While loop:

```
While <condition> do
  { <statements> }
```

→ For loop:

```
for variable := value-1 to value2 step
  do
  { <statements> }
```

→ repeat - until:

```
repeat
{ <statements> }
  until <condition> }
```

* Conditional statements

→ If <condition> then
  <statements>

→ If <condition> then
  <statements>
  else <statements>

→ case

```
{ : <cond-1> : <statement-1>
  .
  .
  .
  : <cond-n> : <statement-n>
  : else : <statement-n+1> }
```

4) Write an algorithm to find factorial of a given number:

1) Start
2) Read n
3) Initialize counter variable i = 1 and fact = 1
4) If i<=n; fact = fact * i ; i++ then to step-4.
   else print fact
5) Stop.

5) Define space complexity and time complexity.

→ Time complexity is a function that describes the taken by the program in terms of quantity of input taken.

→ Space complexity is a function that describes the memory taken by an algorithm in terms of input taken.

6) What is a priority queue?

→ A priority queue is a type of queue that arranges elements based on their priority values.

VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAA LAB

Name K.S.I.SIVANI Roll No. 1602-21-733-052 Page No. 3

PRELAB PROGRAMS : 1

1) Implement the following sorting techniques:

a) Merge Sort:

```
void merge(int arr[], int l, int m, int r)
 { int i,j,k;
  int n1 = m-l+1;
  int n2 = r-m;
  int L[n1], R[n2];
  for(i=0; i<n1; i++)  L[i] = arr[l+i];
  for(j=0; j<n2; j++)  R[j] = arr[m+1+j];
  i=0; j=0; k=l;
 while (i<n1 && j<n2)
  { if (L[i] <= R[j])
      { arr[k] = L[i]; i++;}
    else
      { arr[k] = R[j]; j++; }
    k++ }
```

```
while (i<n1)
  arr[k] = L[i]; i++; k++;}
while (j<n2)
   arr[k] =R[j]; j++; k++;}
}
void mergesort(int arr[], int l, int r)
{ if (l<r)
   { int m= l+(r-l)/2;
     mergesort (arr, l, m);
     mergesort (arr,m+1, r);
     merge (arr, l, m, r); } }.
```

b) QuickSort:

```
void swap (int *a, int *b)
  { int t=*a;
     * a = *b;
     * b = t;}
int partition (int array[], int l, int h)
{  int pivot = array[h];
```

```
    int i=(l-1);
  for(int j=l ; j<h; j++)
  { if(array[j] <=pivot)
     { i++;
       swap(&array[i], &array[j]); } }
       swap (&array[i+1], &array[h]);
        return (i+1); } }
  }
void quickSort (int array[], int l, int h)
 { if(l<h)
   { int pi = partition(array, l, h);
     quickSort(array, l, pi-1);
     quickSort(array, pi+1, h); } }
```

c) Heap Sort:

```
void heapify (int arr[], int n, int i)
{  int largest = i;
   int left = 2*i+1;
    int right = 2*i+2;
   if(left<n && arr[left] > arr[largest])
              largest = left;
   if(right<n && arr[right] > arr[largest])
```

```
        largest = right;
    if (largest != i)
  { swap(&arr[i], &arr[largest]);
    heapify(arr, n, largest); } }
void heapsort(int arr[], int n)
{ for(int i=n/2-1; i>=0; i--)
  { heapify(arr, n, i); }
  for(int i=n-1; i>=0; i--)
    { swap(&arr[0], &arr[i]);
      heapify(arr, i, 0); }
}
```

---

2) WAP to implement binary search:

```
#include <stdio.h>
int binarySearch(int array[], int x, int l, int h)
  { if (h >= l)
    { int mid = l + (h-l)/2;
      if (array[mid] == x)
        return mid;
      if (array[mid] > x)
        return binarySearch(array, x, l, mid-1);
```

* Output:

Element is found at index 2

VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAALAB

Name K.S.T.SIVANI Roll No. 1602-21-733-052 Page No. 7

```
    return binarySearch(array, x, mid+1, h);}
  return -1;}
int main()
{ int array[] = {1, 2, 4, 7, 9, 11, 12};
  int n = sizeof(array) / sizeof(array[0]);
  int x = 4;
  int res = binarySearch(array, x, 0, n-1);
  if (res == -1)
    printf("Not found");
  else
    printf("Element is found at index %d", res);}
```

3) Hashing using linear probing :

```
#include <stdio.h>
#include <stdlib.h>
# define TABLE_SIZE 15
int h[TABLE_SIZE] = {NULL};
void insert()
{ int key, index, i, flag = 0, hkey;
  printf("Enter a value to insert\n");
  scanf("%d", &key);
```

*OUTPUT:

```
1. Insert 2. Display 3. Exit
1
Enter a value to insert
13
1. Insert 2. Display 3. Exit
1
Enter a value to insert
22
1. Insert 2. Display 3. Exit
1
Enter a value to insert
12
1. Insert 2. Display 3. Exit
2
Elements in hashtable:
at index 0 value = 0
at index 1 value = 12
at index 2 value = 13
at index 3 value = 13
at index 4 value = 22
```

VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE
NAME OF THE LABORATORY : DAA LAB
Name K.S.I.SIVANI Roll No. 1602-21-733-052 Page No. 8

```
    hkey = key % TABLE_SIZE;
  for(i=0; i<TABLE_SIZE; i++)
  { index = (hkey + i) % TABLE_SIZE;
    if (h[index] == NULL)
    { h[index] = key;
      break; } }
    if (i == TABLE_SIZE) { printf("Cannot insert"); } }
void display()
{ int i;
  printf("Elements in hashtable:\n");
  for(i=0; i<TABLE_SIZE; i++)
  { printf("\n at index %d \t value = %d", i, h[i]); } }
int main()
{ int opt, i;
  while(1)
  { printf("\n 1. Insert \t 2. Display \t 3. Exit \n");
    scanf("%d", &opt);
    switch(opt)
    { case 1: insert(); break;
      case 2: display(); break;
      case 3: exit(0); } } }
```

* Hashing using separate chaining:

```
#include <stdio.h>
#include <stdlib.h>
#define TABLE_SIZE 10
 struct node
 { int data;
    struct node *next; };
 struct node *head[TABLE_SIZE] ={NULL};*c;
 void insert()
 {int i, key;
   printf("\n Enter a value to insert:\n");
   scanf("%d",&key);
   i= key%TABLE_SIZE;
   struct node *newnode = (struct node *) malloc
   newnode->data = key;          (sizeof (struct node));
   newnode->next = NULL;
   if (head[i]==NULL){ head[i]=newnode;}
   else{ c= head[i];
         while (c->next !=NULL)
          { c=c->next; }
          c-> next = newnode; } }
```

```
void search()
{int key, index;
 printf("\n Enter the element to be searched:\n");
 scanf("%d",&key);
 index = key % TABLE_SIZE;
 if (head[index]==NULL){ printf("Not found\n");}
 else
 {for(c=head[index]; c!= NULL; c=c->next)
   { if(c->data ==key)
     { printf("Search element found\n"); break;}}
 if (c==NULL) printf("Not found\n");}}

void display()
{int i;
 for(i=0; i<TABLE_SIZE; i++)
  {printf(" Entries at index %d\n", i);
   if(head[i]==NULL)
   {printf("No Entry");}
  else
  {for(c=head[i]; c!=NULL; c=c->next)
      printf("%d->", c->data);}}}
```

OUTPUT:

```
Press 1.Insert 2.Display 3.Search 4.Exit
1
Enter a value to insert
12
Press 1.Insert 2.Display 3.Search 4.Exit
1
Enter a value to insert
13
Press 1.Insert 2.Display 3.Search 4.Exit
1
Enter a value to insert
2
Press 1.Insert 2.Display 3.Search 4.Exit
entries at index 0
12 →
entries at index 1
13 →
entries at index 2
2 →
Press 1.Insert 2.Display 3.Search 4.Exit
3
Enter element to be searched
13
Search element found
Press 1.Insert 2.Display 3.Search 4.Exit
4
```

VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAA LAB

Name K.S.I.SIVANI Roll No. 1601-21-733-052 Page No. 11

```
int main()
{ int opt, key, i;
  while(1)
 {printf("\n Press 1.Insert\t 2.Display\t 3.Search\t 4.Exit\n");
  scanf("%d", &opt);
  switch(opt)
  { case 1: insert(); break;
    case 2: display(); break;
    case 3: search(); break;
    case 4: exit(0);
  }
}}
```

VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAA LAB

Name K.S.I. SIVANI Roll No. 1602-21-733-052 Page No. 12

LAB PROGRAMS-1

1) Write a program to implement double hashing:

```
#include <stdio.h>
int hash1(int n)
 {
   return (n%20);
 }
int  hash2(int n)
   { return  (n%13); }
int main()
{ int n, i, j, x, y;
 printf("Enter the no. of elements: ");
 scanf("%d", &n);
 int a[20] = {0};
 for(i=0; i<n; i++)
 { printf("Element:");
   scanf("%d", &x);
   for(j=0; j<20; j++)
   { y = (hash1(x) + j*hash2(x)) %20;
```

*Output:

```
Enter the no. of elements: 11
Element: 16
Element: 8
Element: 63
Element: 9
Element: 27
Element: 37
Element: 48    Collision  Collision.
Element: 5     Collision.
Element: 69
Element: 34
Element: 1
Hashtable:
0 1 0 63 0 5 48 27 8 9 0 0 16 37 0 0.
```

VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.
DEPARTMENT OF : CSE
NAME OF THE LABORATORY : DAA LAB
Name K.S.T.SIVANI Roll No. 1602-21-733-052 Page No. 13

```
if (a[y] == 0)
 { a[y] = x;
   break; }
 else
 { printf("Collision\n");
   continue; }
}}
printf("Hashtable:\n");
for(i=0; i<20; i++)
{ printf("%d\t", a[i]); }
return 0;
 }
```

2) Write a program to implement ternary search:

```
#include <stdio.h>
int tsearch(int a[], int l, int h, int x)
{ int y, z;
   y = l + (h-l)/3;
   z = h - (h-l)/3;
```

```
while(l<=h)
{ if(a[y]==x)
   { return y;}
  else if(a[z]==x)
   { return z;}
  else if(a[y]>x)
   { return tsearch(a,l,y-1,x);}
  else if(a[z]<x)
   { return tsearch(a,z+1,h,x);}
  else
   { return tsearch(a,y+1,z-1,x);}
}
return -1;}
int main()
{ int n,p,res;
  printf("Enter the number of elements:");
  scanf("%d",&n);
  int a[n];
```

* Output:

1) Enter the number of elements: 5
Element: 1
Element: 2
Element: 7
Element: 9
Element: 13
Enter the element to be searched: 7
Found at 2.

2) Enter the number of elements: 6
Element: 1
Element: 3
Element: 5
Element: 7
Element: 9
Element: 11
Enter the element to be searched: 4
Not found.

VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.
DEPARTMENT OF : CSE
NAME OF THE LABORATORY : DAA LAB
Name K.S.I.SIVANI Roll No. 1602-21-733-052 Page No. 15

```
for(int i=0; i<n; i++)
{ printf("Element:");
  scanf("%d", &a[i]);}
printf("Enter the element to be searched:");
scanf("%d", &p);
res = tsearch(a, 0, n, p);
if(res == -1)
{ printf("Not found");}
else
 { printf(" Found at %d", res);}
return 0; }
```