progrqam 1 write a program to implement universal quantification

```
#include <stdio.h>

int main() {
    int x, i;
    printf("The specified domain values of x are:");

    for (i = 0; i < 5; i++) {
        printf("%d ", i);
    }

    printf("\nSelect the integer value of x:");
    scanf("%d", &x);

    if (x >= 0 && x < 5) {
        if (x * x < 10) {
            printf("P(X): x*x < 10 is true for x = %d\n\n", x);
        } else {
            printf("P(X): x*x < 10 is false for x = %d\n\n", x);
        }
    } else {
        printf("The entered x value is not in the specified domain.\n");
    }

    return 0;
}
```

program 2 arthemetic series

```
#include <stdio.h>

int main() {
    float a, d, tn, i, sum = 0.0;
    int n;

    printf("Enter the first number of the A.P. series: ");
    scanf("%f", &a);

    printf("Enter the total numbers in the A.P. series: ");
    scanf("%d", &n);

    printf("Enter the common difference of A.P. series: ");
    scanf("%f", &d);

    tn = a + (n - 1) * d;
    sum = (n * (2 * a + (n - 1) * d)) / 2;

    printf("The given Arithmetic Series is:\n");
    for (i = a; i <= tn; i = i + d) {
        if (i != tn)
            printf("%f\n", i);
        else
            printf("%f\n", i);
    }

    printf("Sum of the given series = %f\n", sum);

    return 0;
}
```

program 3
write a program to fnd the sum of first 10 terms   of sequence 1\n sqaure

```
#include<stdio.h>
#include<math.h>
int main()
{
int i,n=10;
double sl,s2,sum=0.0;
for(i=l;i<=n;i++)
{
printf("n value=&d",i);
s1=pow(i,2);
printf("Itsquare value=%0.5f",s1);
s2=1/s1;
printf("\t\t(1/n2 square value)=%0.5f\n",s2);
sum=sum+s2;
}
printf("in Sum of first 10 terms of the sequence(1/n
sequence)=%0.5f\n",sum);
return 0;
}
#include <stdio.h>
#include <math.h>

int main() {
  int i, n = 10;
  double sl, s2, sum = 0.0;

  for (i = 1; i <= n; i++) {
    sl = pow(i, 2);
    s2 = 1 / sl;
    printf("n value=%d", i);
    printf("Itsquare value=%0.5f", sl);
    printf("\t\t(1/n2 square value)=%0.5f\n", s2);
    sum += s2;
  }

  printf("in Sum of first 10 terms of the sequence(1/n
sequence)=%0.5f\n", sum);

  return 0;
}
```

program 4 :
intersection of two sets

```
#include<stdio.h>
int main()
{
int i,j,k=0,n1,n2,a[10],b[10],c[10];
printf("Enter the Size of first set A:");
scanf("%d",&n1);
printf("Enter the elements of set A:\n");
for(i=0;i<n1;i++)
{
scanf("%d",&a[i]);
```

```
}
printf("Elements of set A are: ");
for(i=0;i<n1;i++)
{
printf("a[%d]=%d\t",i,a[i]);
}
printf("\nmEnter the Size of second set B:");
scanf("*d",&n2);
printf("Enter the elements of set B:\n");
for(j=0;j<n2;j++)
{
scanf("%d",&b[j]);
}
printf( "Elements of set B are:");
for(j=0;j<n2;j++)
{
printf("a[%d]=%d\t",j,b[j]);
}
for(i=0;i<n1;i++)
{
for(j=0;j<n2;j++)
{
if(a[i]==b[j])
{
c[k]-a[i];
k++;
}
}
}
if(k==0)
{
printf("\nIntersection is not possible:");
}
else
{
    printf("\nnIntersection of two sets A and B are: ");
for(i=0;i<k;i++)
{
if(c[i]!=c[i+1])
{
printf("%d\t",c[i]);
return 0;
}
}
}

#include <stdio.h>

int main() {
  int i, j, k = 0, n1, n2, a[10], b[10], c[10];

  printf("Enter the Size of first set A:");
  scanf("%d", &n1);
  printf("Enter the elements of set A:\n");
  for (i = 0; i < n1; i++) {
    scanf("%d", &a[i]);
  }
  printf("Elements of set A are: ");
  for (i = 0; i < n1; i++) {
    printf("a[%d]=%d\t", i, a[i]);
  }

  printf("\nEnter the Size of second set B:");
  scanf("*d", &n2);
  printf("Enter the elements of set B:\n");
  for (j = 0; j < n2; j++) {
    scanf("%d", &b[j]);
  }
  printf("Elements of set B are:");
  for (j = 0; j < n2; j++) {
    printf("a[%d]=%d\t", j, b[j]);
  }

  for (i = 0; i < n1; i++) {
    for (j = 0; j < n2; j++) {
      if (a[i] == b[j]) {
        c[k] = a[i];
        k++;
      }
    }
  }

  if (k == 0) {
    printf("\nIntersection is not possible:");
  } else {
    printf("\nIntersection of two sets A and B are: ");
    for (i = 0; i < k; i++) {
      for (j = i + 1; j < k; j++) {
        if (c[i] == c[j]) {
          for (int k = j; k < k - 1; k--) {
            c[k] = c[k + 1];
          }
          k--;
        }
      }
      printf("%d\t", c[i]);
    }
  }

  return 0;
}
```

program 5
power of given set

```
#include <stdio.h>
#include <math.h>

int main() {
  int n, i, count, power;
  char set[100];

  printf("Enter the size of set:");
  scanf("%d", &n);
  printf("Enter the elements of set :\n");
  for (i = 0; i < n; i++) {
    scanf(" %c", &set[i]);
  }
  power = pow(2, n);
  printf("Number of subsets to be displayed are: %d\n", power);
  for (count = 0; count < power; count++) {
    for (i = 0; i < n; i++) {
      if (count & (1 << i)) {
        printf("%c", set[i]);
      }
    }
    printf("\n");
  }

  return 0;
}
```

program 6
factorial of given number using recursionm

```
#include <stdio.h>

long int fact(int n) {
  if (n == 0 || n == 1) {
    return 1;
  } else if (n > 0) {
    return (n * fact(n - 1));
  }
}

int main() {
  int n, i;
  printf("Enter the number to gets it's factorial:\n");
  scanf("%d", &n);
  if (n < 0) {
    printf("Factorial exists only for Positive number\n");
  } else {
    printf("Factorial of %d : %ld\n", n, fact(n));
  }
  return 0;
}
```

program 7 :
fibinacci suing recursiion

```
#include <stdio.h>

int fib(int);

int main()
{
    int n, i;
    printf("Enter the number of elements you want in the series:\n");
    scanf("%d", &n);
    printf("Fibonacci series is:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d\n", fib(i));
    }
    return 0;
}

int fib(int i)
{
    if (i == 0)
        return 0;
    else if (i == 1)
        return 1;
    else
        return (fib(i - 1) + fib(i - 2));
}
```

program 8
tower of hanoi

```
#include <stdio.h>

void TOH(int n, char from, char aux, char to);
int count = 0;

int main()
{
    int n;
    printf("Enter the number of discs:\n");
    scanf("%d", &n);
    TOH(n, 'A', 'B', 'C');
    printf("In Total number of disc moves = %d\n", count);
    return 0;
}

void TOH(int n, char from, char aux, char to)
{
    if (n == 0)
    {
        return;
    }
    else
    {
        TOH(n - 1, from, to, aux);
        printf("Move disc %d from %c to %c\n", n, from, to);
        count++;
        TOH(n - 1, aux, from, to);
    }
}
```

program 9
binary search tree

```
#include <stdio.h>

int binary_search(int key, int a[], int low, int high);
int main()
{
    int n, i, a[20], key, pos;
    printf("Enter the value of n:\n");
    scanf("%d", &n);
    printf("Enter the %d elements in sorted form:\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    printf("Enter the item to be searched:\n");
    scanf("%d", &key);
    pos = binary_search(key, a, 0, n - 1);
    if (pos == -1)
        printf("Item is not found\n");
    else
        printf("Item is found at %d position\n", pos + 1);
    return 0;
}

int binary_search(int key, int a[], int low, int high)
{
    int mid;
    if (low > high)
        return -1;
    mid = (low + high) / 2;
    if (key == a[mid])
        return mid;
    if (key < a[mid])
        return binary_search(key, a, low, mid - 1);
    else
        return binary_search(key, a, mid + 1, high);
}
```

program 10
merge sort technique

```
#include <stdio.h>

void merge_sort(int a[], int low, int high);
void merge(int a[], int low, int mid, int high);

int main()
{
    int i, n, a[30];
    printf("Enter the array size:\n");
    scanf("%d", &n);
    printf("Enter the elements:\n");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }

    merge_sort(a, 0, n - 1);
    printf("The sorted array is:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d\n", a[i]);
    }
    return 0;
}

void merge_sort(int a[], int low, int high)
{
    int mid;
    if (low < high)
    {
        mid = (low + high) / 2;
        merge_sort(a, low, mid);
        merge_sort(a, mid + 1, high);
        merge(a, low, mid, high);
    }
}

void merge(int a[], int low, int mid, int high)
{
    int i, j, k, c[30];
    i = low;
    j = mid + 1;
    k = low;

    while ((i <= mid) && (j <= high))
    {
        if (a[i] < a[j])
            c[k] = a[i++];
        else
            c[k] = a[j++];
        k++;
    }

    while (i <= mid)
        c[k++] = a[i++];
    while (j <= high)
        c[k++] = a[j++];

    for (i = low; i <= high; i++)
        a[i] = c[i];
}
```

program 11

maximum minimum

```
#include <stdio.h>

int a[50], max, min;

void maxmin(int i, int j, int *max, int *min);

int main()
{
    int i, j, n;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    printf("Enter the elements of the array:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    max = min = a[0];
    maxmin(0, n - 1, &max, &min);
    printf("Minimum = %d\n", min);
    printf("Maximum = %d\n", max);
    return 0;
}

void maxmin(int i, int j, int *max, int *min)
{
    int mid, max1, min1;

    if (i == j)
    {
        *max = *min = a[i];
    }
    else if (i == j - 1)
    {
        if (a[i] > a[j])
        {
            *max = a[i];
            *min = a[j];
        }
        else
        {
            *max = a[j];
            *min = a[i];
        }
    }
    else
    {
        mid = (i + j) / 2;
        maxmin(i, mid, max, min);
        maxmin(mid + 1, j, &max1, &min1);

        if (*max < max1)
            *max = max1;
        if (*min > min1)
            *min = min1;
    }
}


program 12
  binary tree #include <stdio.h>
#include <stdlib.h>

struct node
{
    int value;
    struct node* left;
    struct node* right;
};

struct node* createNode(int data)
{
```

```c
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->value = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

struct node* insertNode(struct node* root, int data)
{
    if (root == NULL)
        return createNode(data);
    if (data < root->value)
        root->left = insertNode(root->left, data);
    else if (data > root->value)
        root->right = insertNode(root->right, data);
    return root;
}

void insertOrder(struct node* root)
{
    if (root == NULL)
        return;
    insertOrder(root->left);
    printf("%d\t", root->value);
    insertOrder(root->right);
}

int main()
{
    struct node* root = NULL;
    root = insertNode(root, 20);
    root = insertNode(root, 5);
    root = insertNode(root, 3);
    root = insertNode(root, 13);
    root = insertNode(root, 25);
    root = insertNode(root, 8);
    root = insertNode(root, 32);
    root = insertNode(root, 6);
    root = insertNode(root, 9);
    root = insertNode(root, 1);
    root = insertNode(root, 15);
    root = insertNode(root, 24);
    root = insertNode(root, 30);
    insertOrder(root);
    return 0;
}
```