|  |  |
| --- | --- |
| **NAME** | Anand Tiwari |
| **UID** | 2021700068 |
| **CLASS** | SY CSE(DS) |
| **BATCH** | D |

Exp1-A

|  |  |
| --- | --- |
| **AIM** | To implement the various functions e.g. linear, non-linear, quadratic, exponential etc. |
| **PROGRAM** | #include<stdio.h>  #include<math.h>    double fac(int n)    {    if (n==0)     return 1;     return n\*fac(n-1);    }    float fun1(int a)    {      return sqrt(a);    }    float fun2(int a)    {      return log(a);    }    float fun3(int a)    {      return log(log(a));    }    float fun4(int a)    {      return pow(sqrt(2),log(a));    }    float fun5(int a)    {      return a;    }    float fun6(int a)    {      return 2\*a+3;;    }    float fun7(int a)    {      return pow(log(a),2);    }    float fun8(int a)    {      return log(fac(a));    }    float fun9(int a)    {      return sqrt(log(a));    }    float fun10(int a)    {      return pow(2,log(a));    }   int main()   {         for ( int i=0;i<=100;i=i+10)      {         printf(" value of %d in function 1 is %0.2f\n",i, fun1(i)); *//*       printf(" value of %d in function 2 is %0.2f\n",i, fun2(i));       printf(" value of %d in function 3 is %0.3f\n",i, fun3(i));       printf(" value of %d in function 4 is %0.2f\n",i, fun4(i));       printf(" value of %d in function 5 is %0.2f\n",i, fun5(i));       printf(" value of %d in function 6 is %0.2f\n",i, fun6(i));       printf(" value of %d in function 7 is %0.2f\n",i, fun7(i));       printf(" value of %d in function 8 is %0.2f\n",i, fun8(i));       printf(" value of %d in function 9 is %0.2f\n",i, fun9(i));       printf(" value of %d in function 10 is %0.2f\n",i, fun10(i));       printf(" value of %d factorial is %0.2f\n",i, fac(i));       printf("\n");        }     } |
| **OUTPUT** |  |
| **GRAPHS** |  |
| **CONCLUSION** | 1 I have created ten functions and try the functions for different value and also draw the graph too understand the behaviour of functions. Every function is a increasing function. Mostly functions are overlapping |

**Exp1-B**

|  |  |
| --- | --- |
| **AIM** | Experiment on finding the running time of an algorithm.(Insertion and Selection sort) |
| **ALGORITHM** | **Insertion sort-**  **Step 1 -** If the element is the first element, assume that it is already sorted. Return 1.  **Step2 -** Pick the next element, and store it separately in a **key.**  **Step3 -** Now, compare the **key** with all elements in the sorted array.  **Step 4 -** If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.  **Step 5 -** Insert the value.  **Step 6 -** Repeat until the array is sorted.  **Selection sort-**  **Step 1** − Set MIN to location 0  **Step 2** − Search the minimum element in the list  **Step 3** − Swap with value at location MIN  **Step 4** − Increment MIN to point to next element  **Step 5** − Repeat until list is sorted |
| **PROGRAM** | #include <stdio.h>  #include <math.h>  #include <conio.h>  #include <stdlib.h>  #include <time.h>  void getInput()  {  *FILE* \*fp;    fp = fopen("input.text","w");    for(int i=0;i<100000;i++)    fprintf(fp,"%d ",rand()%100000);    fclose(fp);  }  void insertionSort(int arr*[]*, int size) {      for (int i = 1; i < size; i++) {          int key = arr[i];          int j = i - 1;          while (key < arr[j] && j >= 0) {              arr[j + 1] = arr[j];              --j;          }          arr[j + 1] = key;      }  }  void selectionSort(int arr*[]*, int len){      int minIndex, temp;      for(int i=0; i<len; i++){          minIndex = i;          for(int j=i+1; j<len; j++){              if(arr[j] < arr[minIndex]){                  minIndex = j;              }          }          temp = arr[minIndex];          arr[minIndex] = arr[i];          arr[i] = temp;      }  }  int main(){      getInput();  *FILE* \*rt, \*tks;      int a=99;      int arrNums[100000];  *clock\_t* t;      rt = fopen("input.text", "r");      tks = fopen("iTimes.txt", "w");      for(int i=0; i<300; i++){          for(int j=0; j<=a; j++){              fscanf(rt, "%d", &arrNums[j]);          }          t = clock();          insertionSort(arrNums, a+1);          t = clock() - t;          double time\_taken = ((double)t)/CLOCKS\_PER\_SEC;          fprintf(tks, "time taken for %d iteration is %Lf\n", (i+1), time\_taken);          printf("%d\t%lf\n", (i+1), time\_taken);          a = a + 100;          fseek(rt, 0, SEEK\_SET);      }      fclose(tks);      tks = fopen("STimes.txt", "w");      a=99;      for(int i=0; i<300; i++){          for(int j=0; j<=a; j++){              fscanf(rt, "%d", &arrNums[j]);          }          t = clock();          selectionSort(arrNums, a+1);          t = clock() - t;          double time\_taken = ((double)t)/CLOCKS\_PER\_SEC;          fprintf(tks, "time taken for %d iteration is %Lf\n", (i+1), time\_taken);          printf("%d\t%lf\n", (i+1), time\_taken);          a = a + 100;          fseek(rt, 0, SEEK\_SET);      }      fclose(tks);      fclose(rt);      return 0;  } |
| **RESULT** | **Selection Sort-**    **Insertion sort-** |
| **GRAPH** |  |
| **CONCLUSION** | I HAVE LEARNT AND IMPLEMENT THE INSERTION AND SELECTION SORT  ALGORITHM AND COMPARE THE TIME COMPLEXITY BETWEEN THESE TWO ALGORITHM.  INSERTION SORT HAS BETTER TIME COMPLEXITY THAN SELECTION |