LAB_2:

MERGE_SORT:

```
#include<bits/stdc++.h>

#include<time.h>

using namespace std;


 void merge(int A[ ] , int start, int mid, int end) {

int p = start ,q = mid+1;


int Arr[end-start+1] , k=0;


for(int i = start ;i <= end ;i++) {

    if(p > mid)

       Arr[k++] = A[q++] ;


   else if ( q > end)

       Arr[k++] = A[p++];


   else if( A[ p ] < A[ q ])

      Arr[k++] = A[p++];


   else

      Arr[k++] = A[q++];

 }

  for (int p=0 ; p< k ;p ++) {


     A[ start++] = Arr[p] ;

  }

}
```

```
 void merge_sort (int A[] , int start , int end )
   {
           if( start < end ) {
           int mid = (start + end ) / 2 ;
           merge_sort (A, start , mid ) ;
           merge_sort (A,mid+1 , end ) ;

          merge(A,start , mid , end );
   }
}
int main()
{
 int array[20];
 srand(time(NULL));
 for(int i=0;i!=20;i++)
 {
 array[i]=rand()%100;
 }
 cout<<"INPUT: ";
 for(int i=0;i!=20;i++)
 {
 cout<<array[i]<<" ";
 }
 merge_sort(array,0,20);
 cout<<endl<<"sorted List: ";
 for(int i=0;i!=20;i++)
 {
 cout<<array[i]<<"   ";
 }
}
```

INPUT: 95   14   78   96   89   70   21   85   9   84   48   91   9   49   51   35   26   25   0   30

sorted List: 0   9   9   14   21   25   26   30   35   48   49   51   70   78   84   85   89   91   95   96

Process returned 0 (0x0)   execution time : 0.019 s

Press any key to continue.

QUICK_SORT:

```
#include<bits/stdc++.h>
#include<time.h>

using namespace std;

int partition ( int A[],int start ,int end) {
    int i = start + 1;
    int piv = A[start] ;
    for(int j =start + 1; j <= end ; j++ )  {

          if ( A[j] < piv) {
                 swap (A[i],A [j]);
            i += 1;
        }
   }
   swap ( A[start] ,A[i-1]) ;
   return i-1;
}

void quick_sort ( int A[ ] ,int start , int end ) {
   if( start < end ) {
        int piv_pos = partition (A,start , end ) ;
        quick_sort (A,start , piv_pos -1);
        quick_sort ( A,piv_pos +1 , end) ;
   }
}

int main()
{
 int array[20];
 srand(time(NULL));
 for(int i=0;i!=20;i++)
 {
 array[i]=rand()%100;
 }
 cout<<"INPUT: ";
 for(int i=0;i!=20;i++)
 {
 cout<<array[i]<<" ";
 }
 quick_sort(array,0,20);
 cout<<endl<<"sorted List: ";
 for(int i=0;i!=20;i++)
 {
 cout<<array[i]<<"   ";
 }
}
```

```
INPUT: 98 85 91 51 77 45 99 81 73 84 94 4 45 80 24 67 70 14 35 92
sorted List: 4   14   24   35   45   45   51   67   70   73   77   80   81   84   85   91   92   94   98   99
Process returned 0 (0x0)   execution time : 0.015 s
Press any key to continue.
```

KNAPSACK PROBLEM:

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

void knapsack(int n, float weight[], float profit[], float capacity) {
  float x[20], tp = 0;
  int i, j, u;
  u = capacity;

  for (i = 0; i < n; i++)
    x[i] = 0.0;

  for (i = 0; i < n; i++) {
    if (weight[i] > u)
      break;
    else {
      x[i] = 1.0;
      tp = tp + profit[i];
      u = u - weight[i];
    }
  }

  if (i < n)
    x[i] = u / weight[i];

  tp = tp + (x[i] * profit[i]);

  cout<<"Maximum profit is : "<<tp<<endl;

}

int main() {
  int capacity=35;
  int num=5, i, j;
  float ratio[20], temp;
  float weight[5]={15, 12, 20, 18, 10};
  float profit[5]={20, 25, 15, 28, 22};



  for (i = 0; i < num; i++) {
    ratio[i] = profit[i] / weight[i];
  }



  for (i = 0; i < num; i++) {
    for (j = i + 1; j < num; j++) {
      if (ratio[i] < ratio[j]) {
        temp = ratio[j];
```

```
            ratio[j] = ratio[i];
            ratio[i] = temp;

            temp = weight[j];
            weight[j] = weight[i];
            weight[i] = temp;

            temp = profit[j];
            profit[j] = profit[i];
            profit[i] = temp;
         }
      }
   }

   knapsack(num, weight, profit, capacity);
}
```

Input

n=5, W = 35, (p1, p2, p3, p4, p5)=(20, 25, 15, 28, 22) and (w1, w2, w3, w4, w5) = (15, 12, 20, 18, 10).

Output

```
Maximum profit is : 67.2222

Process returned 0 (0x0)   execution time : 0.013 s

Press any key to continue.
```