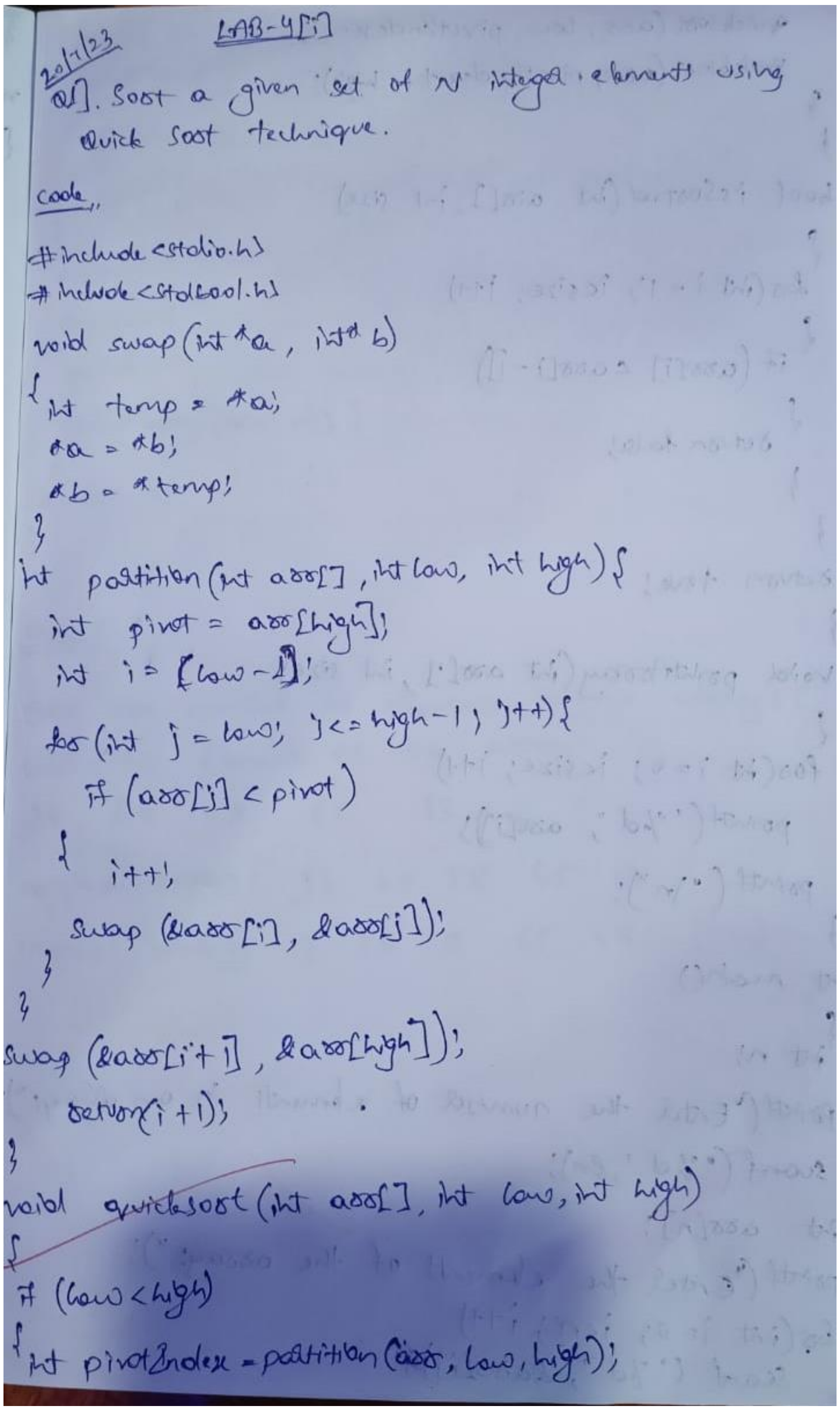
LAB 4

EXP NO: 5

Sort a given set of N integer elements using quick sort technique.



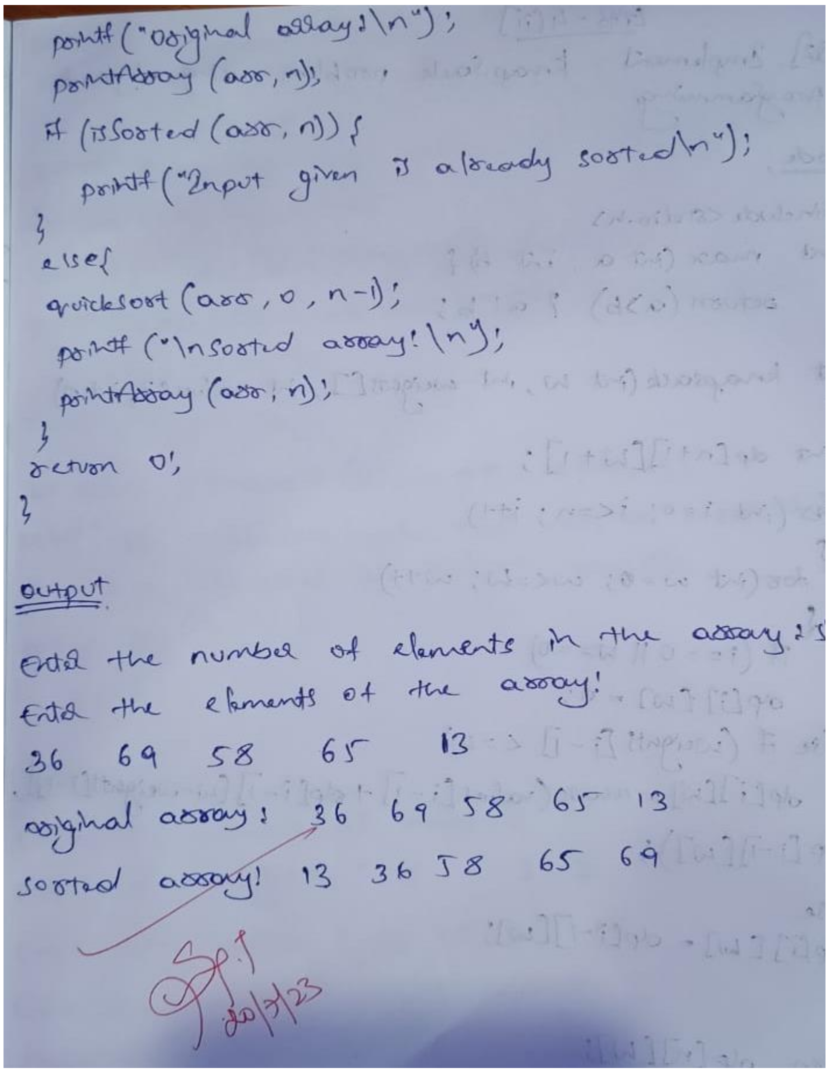
20/1/23 LAB-4[i]

Q1]. Sort a given set of N integer elements using Quick Sort technique.

code,,

```
#include<stdio.h>
#include<stdbool.h>
void swap(int *a, int* b)
{ int temp = *a;
  *a = *b;
  *b = *temp;
}
int partition(int arr[], int low, int high){
 int pivot = arr[high];
 int i = (low-1);
 for(int j = low; j<= high-1; j++){
    if (arr[j] < pivot)
   {
      i++;
    swap (&arr[i], &arr[j]);
   }
 }
swap (&arr[i+1], &arr[high]);
  return(i+1);
}
void quicksort(int arr[], int low, int high)
{
if (low<high)
{int pivotIndex = partition(arr, low, high);
```

```
  quicksort (arr, low, pivotIndex-1);
   quicksort (arr, pivotIndex+1, high);
 }
}
bool isSorted (int arr[], int size)
 {
  for(int i = 1; i<size; i++)
  {
   if (arr[i] <arr[i-1])
   {
    return false;
   }
 }
return true;
}
void printArray(int arr[], int size)
 {
  for(int i=0; i<size; i++)
    printf("%d ", arr[i]);
   printf("\n");
 }
int main()
{
 int n;
 printf("Enter the number of elements in an array:")
  scanf("%d", &n);
 int arr[n];
printf("Enter the elements of the array: ");
for(int i=0; i<n; i++)
  scanf ("%d", &arr[i]);
```



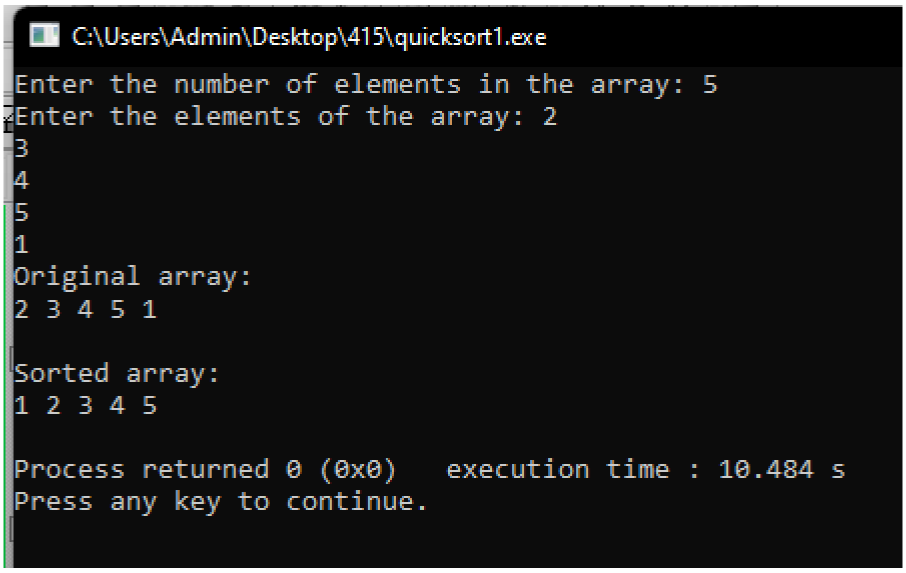
```
printf("original array:\n");
printArray(arr, n);
if (isSorted(arr, n)) {
    printf("Input given is already sorted\n");
}
else{
 quicksort(arr, 0, n-1);
 printf("\nSorted array:\n");
 printArray(arr, n);
}
return 0;
}
```

Output

```
Enter the number of elements in the array: 5
Enter the elements of the array!
36  69  58  65  13
original array: 36  69  58  65  13
sorted array! 13  36  58  65  69
```

[illegible] 20/3/23

Output:



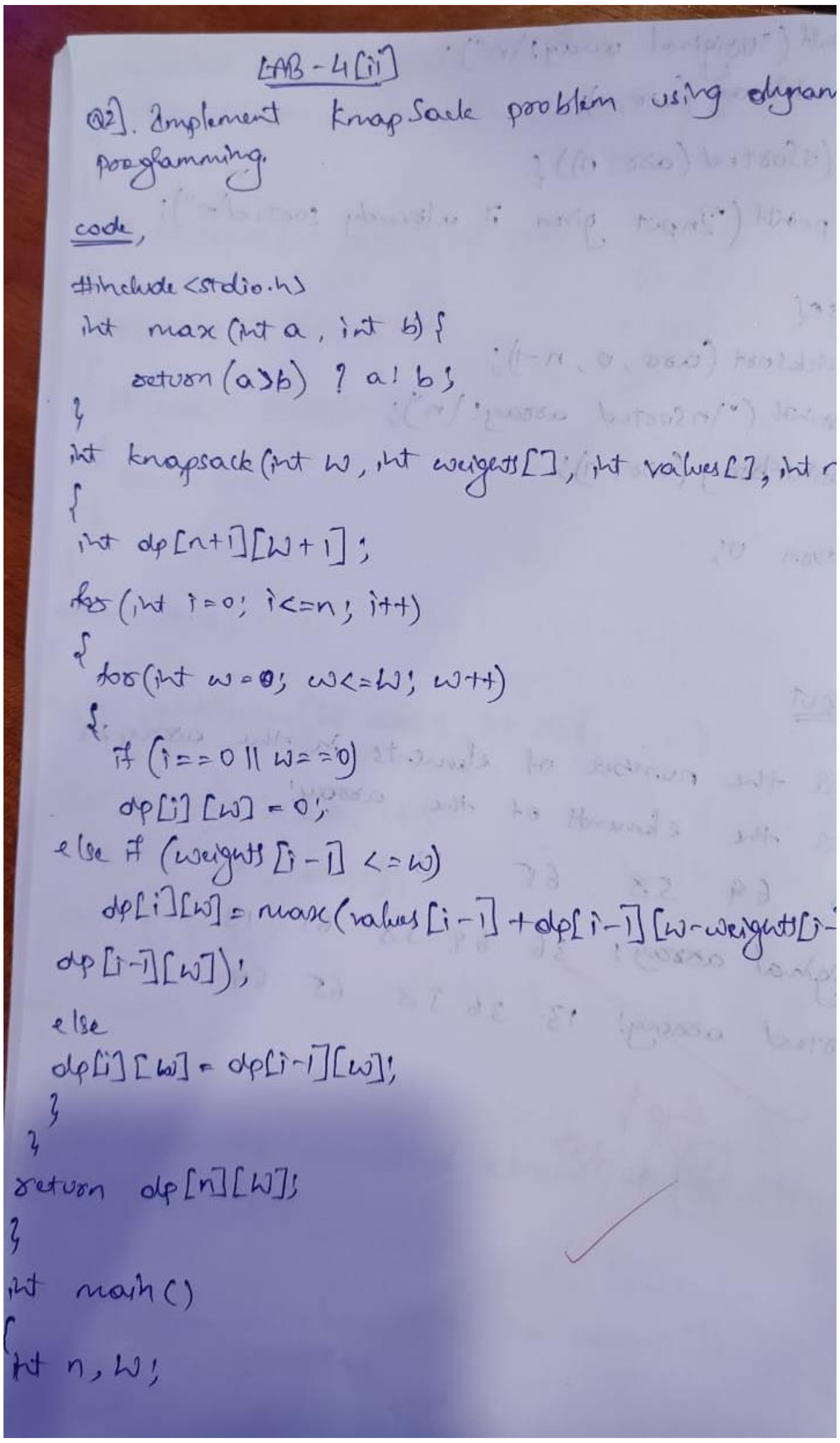
```
C:\Users\Admin\Desktop\415\quicksort1.exe
Enter the number of elements in the array: 5
Enter the elements of the array: 2
3
4
5
1
Original array:
2 3 4 5 1

Sorted array:
1 2 3 4 5

Process returned 0 (0x0)   execution time : 10.484 s
Press any key to continue.
```

EXP NO : 6

Implement knapsack problem using dynamic programming.

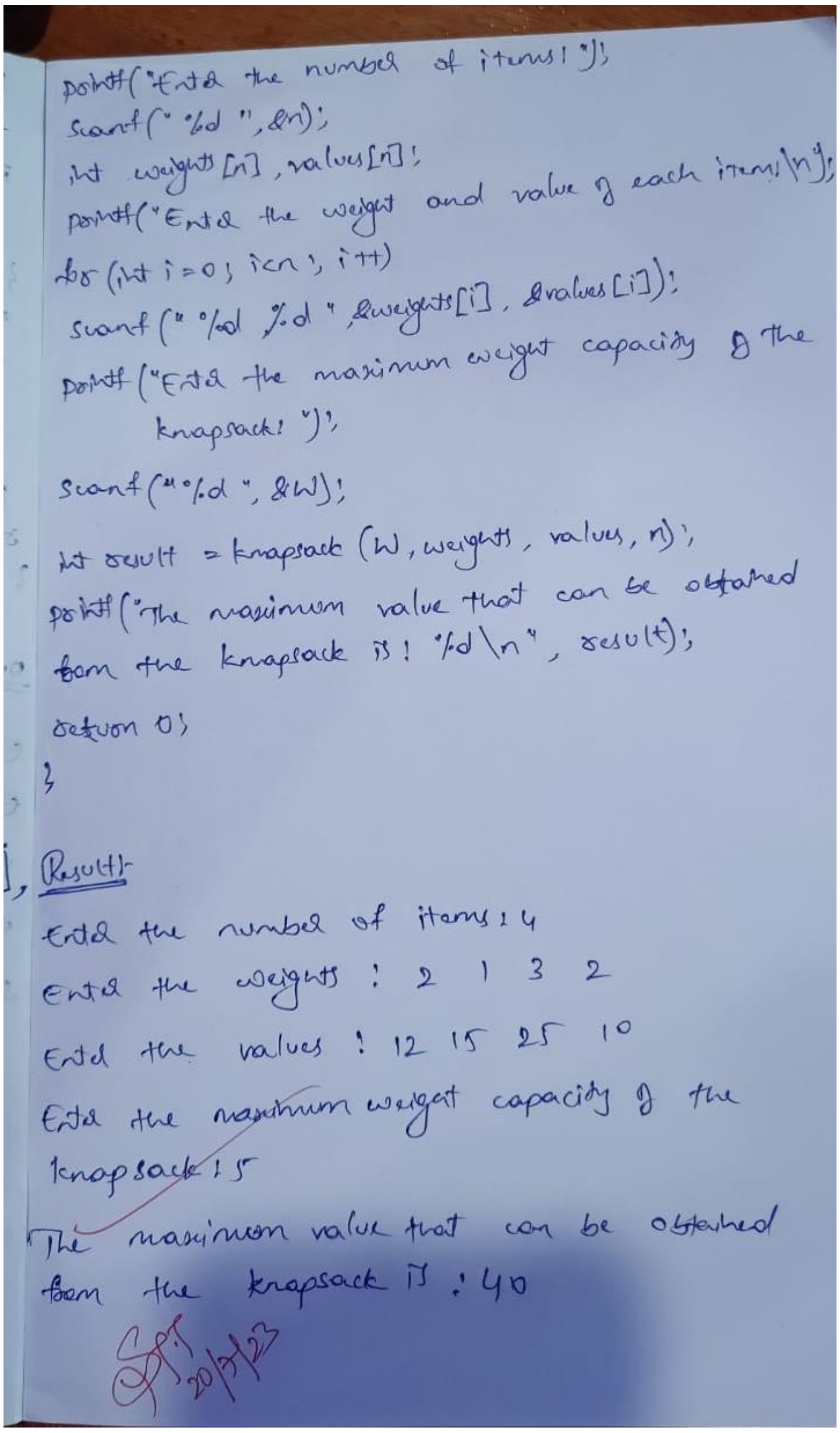


LAB - 4 [ii]

Q2]. Implement KnapSack problem using dynam
programming.

code,

```
#include <stdio.h>
int max (int a, int b) {
    return (a>b) ? a : b;
}
int knapsack (int W, int weights[], int values[], int n
{
int dp[n+1][W+1];
for (int i=0; i<=n; i++)
{ for(int w=0; w<=W; w++)
 {
   if (i==0 || w==0)
   dp[i][w] = 0;
else if (weights[i-1] <=w)
   dp[i][w] = max(values[i-1] + dp[i-1][w-weights[i-
dp[i-1][w]);
else
dp[i][w] = dp[i-1][w];
 }
}
return dp[n][W];
}
int main()
{
int n, W;
```



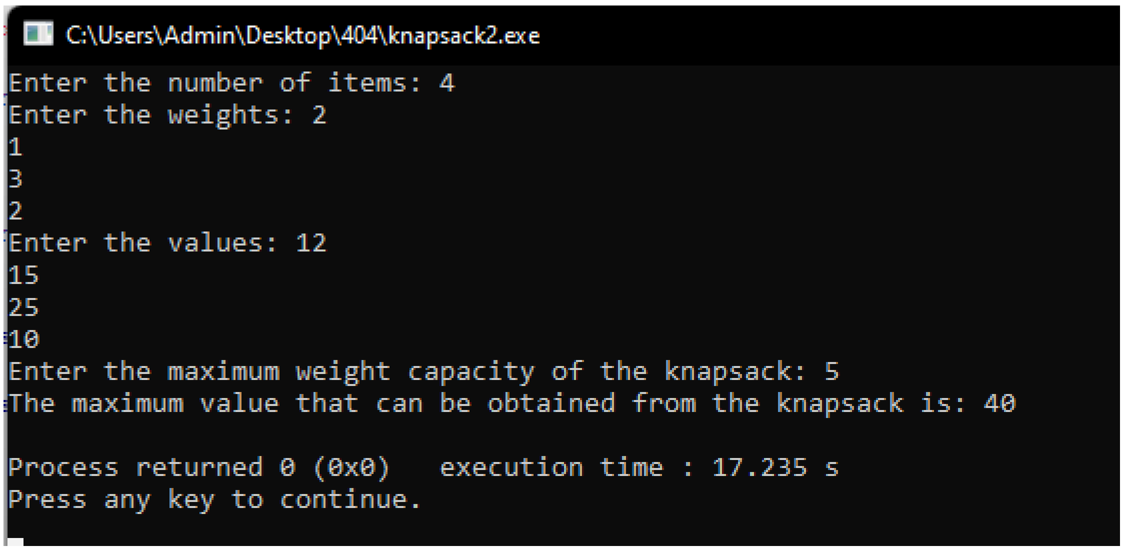
```
printf("Enter the number of items: ");
scanf(" %d ", &n);
int weights [n], values[n];
printf("Enter the weight and value of each item:\n");
for (int i=0; i<n; i++)
 scanf(" %d %d ", &weights[i], &values[i]);
printf ("Enter the maximum weight capacity of the
        knapsack: ");
scanf(" %d ", &W);
int result = knapsack (W, weights, values, n);
printf("The maximum value that can be obtained
from the knapsack is: %d\n", result);
return 0;
}
```

Result:-

```
Enter the number of items: 4
Enter the weights : 2 1 3 2
Enter the values : 12 15 25 10
Enter the maximum weight capacity of the
knapsack: 5
The maximum value that can be obtained
from the knapsack is : 40
```

[illegible] 20/7/23

Output:



```
Enter the number of items: 4
Enter the weights: 2
1
3
2
Enter the values: 12
15
25
10
Enter the maximum weight capacity of the knapsack: 5
The maximum value that can be obtained from the knapsack is: 40

Process returned 0 (0x0)   execution time : 17.235 s
Press any key to continue.
```