CSE320 Lab report .C

Quick Sort Result

```
[006704029@csusb.edu@csevnc lan1C]$ g++ qsc.c
[006704029@csusb.edu@csevnc lan1C]$ ./a.out
unsorted_array:
11 22 44 33 77 66 55 99
run_time: 0.000071
sorted_array:
11 22 33 44 55 66 77 99

unsorted_array:
112 28 81 198 92 466 58 597 46 989
run_time: 0.000018
sorted_array:
28 46 58 81 92 112 198 466 597 989

unsorted_array:
222 444 587 456 455 678 716 729 782 239 195 495 794 309 988
run_time: 0.000017
sorted_array:
195 222 239 309 444 455 456 495 587 678 716 729 782 794 988
```

Partition Result:

```
[006704029@csusb.edu@csevnc lan1C]$ g++ partic.c
[006704029@csusb.edu@csevnc lan1C]$ ./a.out
Divided[006704029@csusb.edu@csevnc lan1C]$ g++ partic.c
[006704029@csusb.edu@csevnc lan1C]$ ./a.out
Divided[006704029@csusb.edu@csevnc lan1C]$ g++ partic.c
[006704029@csusb.edu@csevnc lan1C]$ ./a.out
Divided[006704029@csusb.edu@csevnc lan1C]$
```

Source Code

```c
#include <stdio.h>
#include <time.h>

int partition(int array[], int l, int h){
        int pivot = array[h];
        int i = (l - 1);
        for(int j = l; j <= h - 1; j++){
                if(array[j] < pivot){
                        i++;
                        int temp = array[i];
                        array[i] = array[j];
                        array[j] = temp;
                }
        }
        int temp = array[i+1];
        array[i+1] = array[h];
        array[h] = temp;
        return (i+1);
}

void quicksort(int array[], int l, int h){
        if(l < h){
                int partitionindex = partition(array, l, h);
                quicksort(array, l, partitionindex - 1);
                quicksort(array, partitionindex + 1, h);
        }
}

void printquicksort(int array[], int arraysize){
        clock_t t;
        t = clock();
        printf("unsorted_array:\n");
        for(int i = 0; i < arraysize; i++)
                printf("%d ", array[i]);
        printf("\n");
        quicksort(array, 0, arraysize-1);
        t = clock() - t;
        double ft = ((double)t)/CLOCKS_PER_SEC;
        printf("run_time: %lf\n", ft);
        printf("sorted_array:\n");
        for(int i = 0; i < arraysize; i++)
                printf("%d ", array[i]);
        printf("\n\n");
}
```

```c
int main(){
       int array1[] = {11, 22, 44, 33, 77, 66, 55,99};
       int arraysize1 = sizeof(array1)/sizeof(array1[0]);
       int array2[] = {112, 28, 81, 198, 92, 466, 58, 597, 46, 989};
       int arraysize2 = sizeof(array2)/sizeof(array2[0]);
       int array3[] = {222, 444, 587, 456, 455, 678, 716, 729, 782, 239, 195, 495, 794, 309, 988};
       int arraysize3 = sizeof(array3)/sizeof(array3[0]);
       printquicksort(array1, arraysize1);
       printquicksort(array2, arraysize2);
       printquicksort(array3, arraysize3);
}
```

```c
#include <stdio.h>
#include <stdbool.h>

bool isSubsetSum (int arr[], int n, int sum)
{

   if (sum == 0)
     return true;
   if (n == 0 && sum != 0)
     return false;

   if (arr[n-1] > sum)
     return isSubsetSum (arr, n-1, sum);

   return isSubsetSum (arr, n-1, sum) ||
          isSubsetSum (arr, n-1, sum-arr[n-1]);
}

bool findPartiion (int arr[], int n)
{

    int sum = 0;
    for (int i = 0; i < n; i++)
       sum += arr[i];

    if (sum%2 != 0)
       return false;

    return isSubsetSum (arr, n, sum/2);
}


int main()
{
        int arr[] = {11,22,33};
  int n = sizeof(arr)/sizeof(arr[0]);
  if (findPartiion(arr, n) == true)
  if (findPartiion(arr, n) == true)
     printf("Divided");
  else
     printf("Can't be divided");
  return 0;
}
```

How easy/hard was it was to program?

C is really similar to C++, therefore this really isn't hard to program at all.

The ease/difficulty of debugging:

Really easy to debug.

The speed of execution:

Run time already shown in the screenshot above