Assignment - 3

```
#include <stdio.h>
#include <stdlib.h>
Struct internal // representing internals in form of structs
{ float start;
  float end;
};
typedef struct internal I;
I *getInternals(int n)
{ I* internals = (I*)(malloc(n*sizeof(I)));
  srand(time(0)) // using randomness to generate internals
  for(int i=0; i<n; i++)
  { float ed = ((float) rand()/(RAND_MAX));
    float st = ((float) rand()/(RAND_MAX));
    internals[i].start = st;
    internals[i].end = ed;
  } /* getting the endpoint of an internal and
  return internals; startpoint of an internal using rand() */
}
void merge(I *arr, int l, int mid, int r) // merging two sorted arrays left and right
int n1 = mid - l + 1;
  int n2 = r - mid;
```

Rishi

```
I* left= (I*) (malloc (n1 * sizeof(I) ));
I* right = (I*)(malloc(n2 * sizeof (I) ));
for(int i=0; i < n1; i++) // declaring a left array of
{ left[i] = arr[l+i]; }                    structs
for(int j=0; j<n2; j++) { right[j] = arr( mid + j + 1]};
int i=0, j=0, k=l;          // declaring a right array
while( i<n1 && j<n2)                of structs
{ // sorting by their start lines
  if( left[i]. start < right[j]. start ||
      left[i]. start == right[j]. start && left[i]. end <
      right[j]. end)
  { arr[k] = left[i];
    i++;
  }
  else
  { arr[k] = right[j];
    j++;
  }
  k++;
}
while(i<n1) // performing the logic for merge
{ arr(k) = left[i];          operations in merge sort
  k++; i++; }
while(j<n2)
{ arr[k]= right[j];
  k++; j++; }
```

int count = 1

```
//performing merge sort operations to sort the array of structs by their start time
void mergeSort(I *arr, int l, int r)
{ if(l<r)
  { int mid = l+(r-l)/2;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
  }
}
int main()
{ int n;
  scanf("%d", &n);
  I *arr = getIntervals(n);
  for(int i=0; i<n; i++) //printing the intervals as generated by random function
  { printf("Start: %lld End: %lld\n", arr[i].start, arr[i].end);
  }
  mergeSort(arr, 0, n-1);
  printf("The sorted list of Intervals are:\n");
  for(int i=0; i<n; i++) //printing the sorted array of structs
  { printf("Start: %lld End: %lld\n", arr[i].start, arr[i].end);

float rs=0; float re=0;
//implementing the required logic
while(j<k)
{ while(i<j && arr[i].start > arr[j].end)
  { i++;
  }
  int count = j-i+1;
```

```
if(j<n && count > res)
{ res = count; // storing the count of overlapped intervals
  rs = arr[j].start; // storing the leftmost part of rectangle
  re = minEnd; // storing the rightmost part of rectangle
  minEnd = arr[j].end;
}
if(minEnd > arr[j].end) // storing the minimum end
   minEnd = arr[j].end;
j++;
}//printing the answer
printf("%d \n", res);
printf("%lf %lf \n", rs, re);
```