NAME – ANSH GOEL

REGISTER NO.- 20BCE1798

COURSE CODE – CSE4001

COURSE NAME – PARALLEL & DISTRIBUTED COMPUTING

SLOT – L9+L10

LAB-3

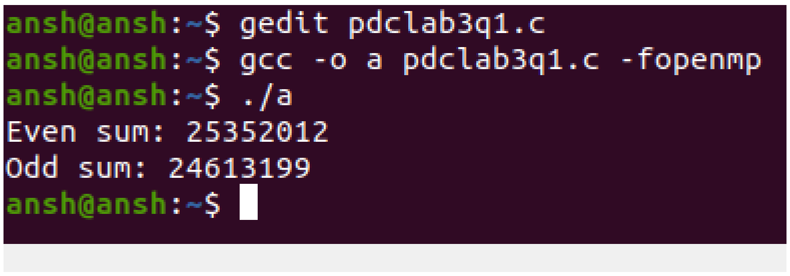
1. Write a parallel program in open MP to create 8, 16 and 32 threads using runtime library routines. Construct an array of 10000 elements. Distribute the loop iterations to 32, 64, 128 concurrent threads with a chunk-size of 10, 20 and 50 using static, dynamic, guided and auto scheduling schemes. Find out the odd and even numbers global sum of 10K items. Record your execution times for the abovementioned schemes.

CODE:

```
#include<stdio.h>
#include<omp.h>
#include<stdlib.h>
#include<time.h>
#define chunksize 10
#define thrdnum 8

int main(){
	int arr[10000];
	int esum=0, osum=0;
	#pragma omp for
	for(int i=0;i<10000;i++){
		arr[i]=rand()%10000;
	}

	#pragma omp for schedule(static,chunksize)
	for(int i=0;i<10000;i++){
		if(arr[i]%2==1)
			osum+=arr[i];
		else
			esum+=arr[i];
```

```
        }
        printf("Even sum: %d\n",esum);
        printf("Odd sum: %d\n",osum);
}
```

OUTPUT:



```
ansh@ansh:~$ gedit pdclab3q1.c
ansh@ansh:~$ gcc -o a pdclab3q1.c -fopenmp
ansh@ansh:~$ ./a
Even sum: 25352012
Odd sum: 24613199
ansh@ansh:~$ 
```

2. Write a parallel program to sort N elements in an array using OpenMP
   i. Bubble Sort
   ii. Quick Sort

CODE:

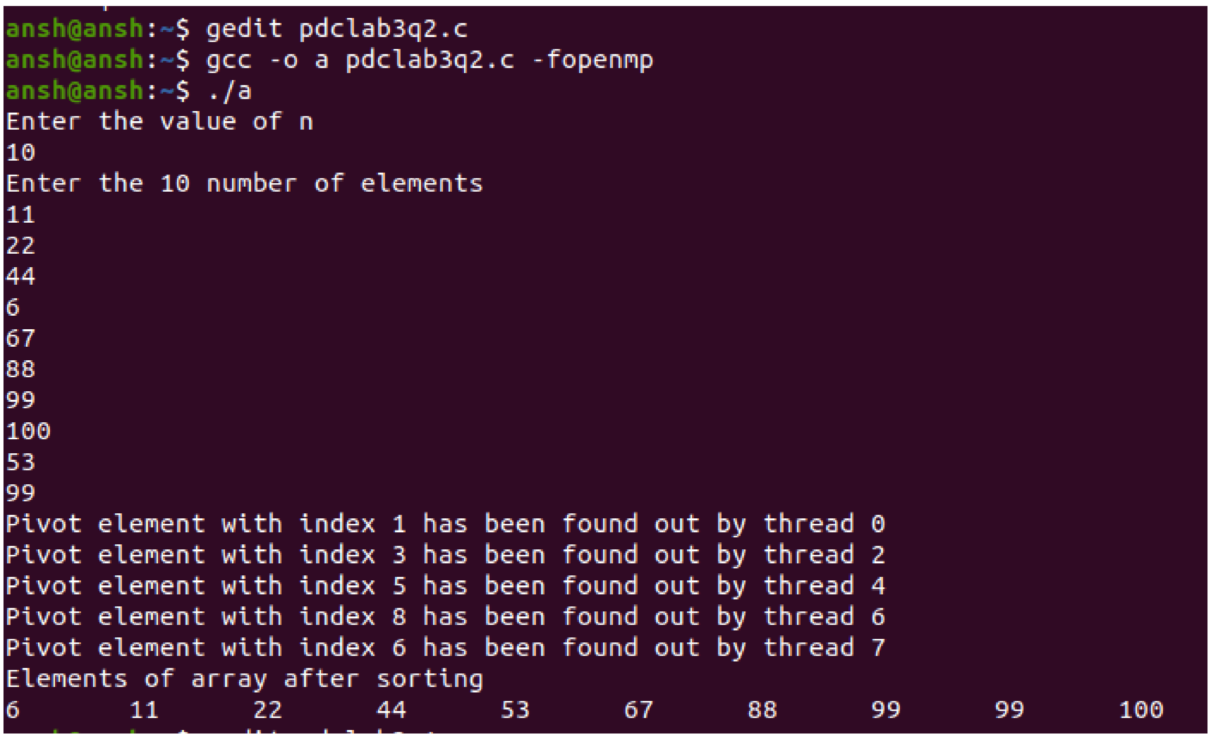
```
#include<stdio.h>
#include<omp.h>

int k=0;

int partition(int arr[], int low_index, int high_index)
{
        int i, j, temp, key;
        key = arr[low_index];
        i= low_index + 1;
        j= high_index;
        while(1)
        {
                while(i < high_index && key >= arr[i])
                        i++;
                while(key < arr[j])
                        j--;
                if(i < j)
```

```
		{
			temp = arr[i];
			arr[i] = arr[j];
			arr[j] = temp;
		}
		else
		{
			temp= arr[low_index];
			arr[low_index] = arr[j];
			arr[j]= temp;
			return(j);
		}
	}
}


void quicksort(int arr[], int low_index, int high_index)
{
	int j;

	if(low_index < high_index)
	{
		j = partition(arr, low_index, high_index);
		printf("Pivot element with index %d has been found out by thread %d\n",j,k);

		#pragma omp parallel sections
		{
			#pragma omp section
			{
				k=k+1;
				quicksort(arr, low_index, j - 1);
			}

			#pragma omp section
```

```
                {
                        k=k+1;
                        quicksort(arr, j + 1, high_index);
                }

        }
    }
}


int main()
{
    int arr[100];
    int n,i;

    printf("Enter the value of n\n");
    scanf("%d",&n);
    printf("Enter the %d number of elements \n",n);

    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }

    quicksort(arr, 0, n - 1);

    printf("Elements of array after sorting \n");

    for(i=0;i<n;i++)
    {
        printf("%d\t",arr[i]);
    }

    printf("\n");
```

}

OUTPUT:



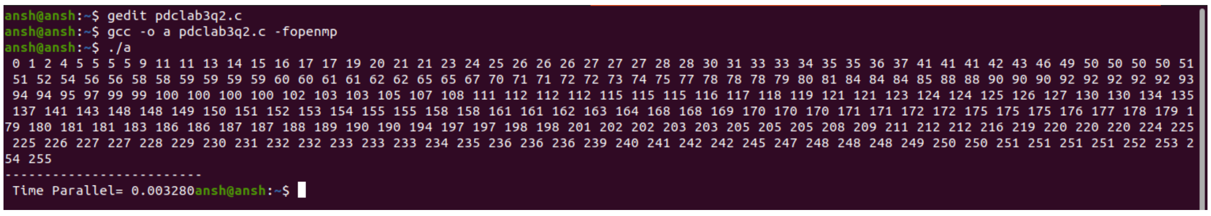
```
ansh@ansh:~$ gedit pdclab3q2.c
ansh@ansh:~$ gcc -o a pdclab3q2.c -fopenmp
ansh@ansh:~$ ./a
Enter the value of n
10
Enter the 10 number of elements
11
22
44
6
67
88
99
100
53
99
Pivot element with index 1 has been found out by thread 0
Pivot element with index 3 has been found out by thread 2
Pivot element with index 5 has been found out by thread 4
Pivot element with index 8 has been found out by thread 6
Pivot element with index 6 has been found out by thread 7
Elements of array after sorting
6       11      22      44      53      67      88      99      99      100
```

ii)

CODE:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

void swap();

int main (int argc, char *argv[]) {
        int SIZE =1<<8;
        int A[SIZE];
        for(int i=0;i<SIZE;i++)
        {
                A[i]=rand()%SIZE;
        }
        int N = SIZE;
        int i=0, j=0;
        int first;
        double start,end;
        start=omp_get_wtime();
        for( i = 0; i < N-1; i++ )
        {
                first = i % 2;
                #pragma omp parallel for default(none),shared(A,first,N)
                for( j = first; j < N-1; j += 1 )
                {
                        if( A[ j ] > A[ j+1 ] )
                        {
                                swap( &A[ j ], &A[ j+1 ] );
                        }
                }
        }
        end=omp_get_wtime();
        for(i=0;i<N;i++)
        {
                printf(" %d",A[i]);
        }

        printf("\n-------------------------\n Time Parallel= %f",(end-start));
}

void swap(int *num1, int *num2)
{

        int temp = *num1;
        *num1 =  *num2;
        *num2 = temp;
}
```

OUTPUT:



```
ansh@ansh:~$ gedit pdclab3q2.c
ansh@ansh:~$ gcc -o a pdclab3q2.c -fopenmp
ansh@ansh:~$ ./a
 0 1 2 4 5 5 5 5 9 11 11 13 14 15 16 17 17 19 20 21 21 23 24 25 26 26 26 27 27 27 28 28 30 31 33 33 34 35 35 36 37 41 41 41 42 43 46 49 50 50 50 50 51
 51 52 54 56 56 58 58 59 59 59 59 60 60 61 61 62 62 65 65 67 70 71 71 72 72 72 73 74 75 77 78 78 78 79 80 81 84 84 84 85 88 88 90 90 90 92 92 92 92 92 93
 94 94 95 97 99 99 100 100 100 100 102 103 103 105 107 108 111 112 112 112 115 115 115 116 117 118 119 121 121 123 124 124 125 126 127 130 130 134 135
 137 141 143 148 148 149 150 151 152 153 154 155 155 158 158 161 161 162 163 164 168 168 169 170 170 170 171 171 172 172 175 175 175 176 177 178 179 1
79 180 181 181 183 186 186 187 187 188 189 190 190 194 197 197 198 198 201 202 202 203 203 205 205 205 208 209 211 212 212 216 219 220 220 220 224 225
 225 226 227 227 228 229 230 231 232 232 233 233 233 234 235 236 236 236 239 240 241 242 242 245 247 248 248 248 249 250 250 251 251 251 251 252 253 2
54 255
-------------------------
 Time Parallel= 0.003280ansh@ansh:~$ 
```