NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY
IT 301 Parallel Computing
14th September 2021
Lab 6

NAME: BHUVANESWAR D
ROLL NO: 191IT107

---

(1) Execute following code and observe the working of task directive.

Check the result by removing if() clause with task. [1+1 = 2 Marks]

```
#include<stdio.h>
#include<omp.h>
int fibo(int n);
int main(void)
{
int n,fib;
double t1,t2;
printf("Enter the value of n:\n");
scanf("%d",&n);
t1=omp_get_wtime();
#pragma omp parallel shared(n)
{
#pragma omp single
{
fib=fibo(n);
}
}
t2=omp_get_wtime();
printf("Fib is %d\n",fib);
printf("Time taken is %f s \n",t2-t1);
return 0;
}

int fibo(int n)
{
int a,b;
if(n<2)
return n;
else
{
#pragma omp task shared(a) if(n>5)
{
printf("Task Created by Thread %d\n",omp_get_thread_num());
a=fibo(n-1);
printf("Task Executed by Thread %d \ta=%d\n",omp_get_thread_num(),a);
}
#pragma omp task shared(b) if(n>5)
{
printf("Task Created by Thread %d\n",omp_get_thread_num());
b=fibo(n-2);
printf("Task Executed by Thread %d \tb=%d\n",omp_get_thread_num(),b);
```

```
}
#pragma omp taskwait
return a+b;
}
```

WITH IF CLAUSE

```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o q1 -fopenmp q1.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./q1
Enter the value of n:
4
Task Created by Thread 3
Task Created by Thread 3
Task Created by Thread 3
Task Executed by Thread 3       a=1
Task Created by Thread 3
Task Executed by Thread 3       b=0
Task Executed by Thread 3       a=1
Task Created by Thread 3
Task Executed by Thread 3       b=1
Task Executed by Thread 3       a=2
Task Created by Thread 3
Task Created by Thread 3
Task Executed by Thread 3       a=1
Task Created by Thread 3
Task Executed by Thread 3       b=0
Task Executed by Thread 3       b=1
Fib is 3
Time taken is 0.000637 s
```

WITHOUT IF CLAUSE

```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o q1 -fopenmp q1.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./q1
Enter the value of n:
4
Task Created by Thread 4
Task Created by Thread 7
Task Created by Thread 6
Task Executed by Thread 6       a=1
Task Created by Thread 5
Task Created by Thread 1
Task Executed by Thread 5       b=1
Task Created by Thread 0
Task Created by Thread 6
Task Executed by Thread 6       a=1
Task Executed by Thread 1       b=0
Task Executed by Thread 7       b=1
Task Created by Thread 2
Task Executed by Thread 2       b=0
Task Executed by Thread 0       a=1
Task Executed by Thread 4       a=2
Fib is 3
Time taken is 0.001044 s
```

EXPLAINATION:Task scheduling is done for values n<4 with if() clause. But without if() clause it is done for all values.

(2) Design a parallel program to find a given element in an unsorted array using Binary Search. Take a large number of elements up to the maximum possible size. Make use of openmp task directive. Use random function to initialise values.

Compare the time taken for searching an element in best case , worst case and average case.

(i) Sequential Binary Search program **[3 Marks]**

(ii) parallel binary search program **[5 Marks]**

BEST CASE

```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o q2 -fopenmp q2.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./q2
Enter the number of elements: 456
        Searching for: 251
TIME TAKEN FOR SEQUENTIAL BINARY SEARCH: 0.000000
        Element found at: 227
TIME TAKEN FOR PARALLEL BINARY SEARCH: 0.000001
        Element found at: 227
```

WORST CASE

```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o q2 -fopenmp q2.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./q2
Enter the number of elements: 456
        Searching for: 465
Sequential Binary Search: 0.000000
        Element not found
 Binary Search: 0.000462
        Element not found
```

AVERAGE CASE

```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o q2 -fopenmp q2.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./q2
Enter the number of elements: 456
        Searching for: 342
Sequential Binary Search: 0.000000
        Element found at: 341
 Binary Search: 0.000402
        Element found at: 341
```

EXPLAINATION:

Sequential execution is faster than parallel since the thread operations are executed in parallel.

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <sys/time.h>
#include <time.h>

void merge(int A[], int l, int m, int r)
{
    int i, j, k;
    int left = m - l + 1;    int right = r - m;
    int L[left], R[right];

    for (i = 0; i < left; i++){     L[i] = A[l + i];}
    for (j = 0; j < right; j++){    R[j] = A[m + 1 + j];}

    i = 0;    j = 0;    k = l;

    while (i < left && j < right)
    {
        if (L[i] <= R[j]){   A[k] = L[i];    i++;}
        else{   A[k] = R[j];    j++;}
        k++;
    }
    while (i < left)
    {   A[k] = L[i];    i++;    k++;}
    while (j < right)
    {   A[k] = R[j];    j++;    k++;}
}

void mergesort(int A[], int l, int r)
{
    if (l < r)
    {
        int mid = l + (r - l) / 2;
        mergesort(A, l, mid);
        mergesort(A, mid + 1, r);
        merge(A, l, mid, r);
    }
}


int parallel(int A[], int l, int r, int element)
{
    if (l > r)
    {   return -1;}
    if (l == r)
    {
```

```
48          if (A[l] == element)
49          {   return l;}
50          return -1;
51      }
52      int mid = l + (r-l)/2;
53
54      if (A[mid] == element)
55      {   return mid;}
56      int result = -1;
57
58      #pragma omp parallel num_threads(4)
59      {
60          #pragma omp single
61          {
62              if (A[mid] > element)
63              {
64                  #pragma omp task
65                  {   result = parallel(A, l, mid-1, element);}
66              }
67              else
68              {
69                  #pragma omp task
70                  {   result = parallel(A, mid+1, r, element);}
71              }
72          #pragma omp taskwait
73          }
74      }
75      return result;
76  }
77
78  int seq(int A[], int n, int element)
79  {
80      int l = 0, r = n-1, mid;
81      while (l < r)
82      {
83          mid = l + (r-l)/2;
84          if (A[mid] == element){   return mid;}
85          if (A[mid] > element){   r = mid-1;}
86          else{   l = mid+1;}
87      }
88      if (A[l] == element)
89      {   return l;}
90      return -1;
91  }
92
93  double timecalc(struct timeval TimeValue_Start,struct timeval TimeValue_Final)
94  {
```

```c
double timecalc(struct timeval TimeValue_Start,struct timeval TimeValue_Final)
{
    long start, end;    double time_total;
    start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_total = (end - start)/1000000.0;
    return time_total;
}

int main()
{
    srand(time(0));
    int n;

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

    int A[n];
    for (int i = 0; i < n; i++) {   A[i] = rand()%n;}
    mergesort(A, 0, n-1);

    int element = A[342], result;
    printf("\tSearching for: %d\n", element);

    struct timeval TimeValue_Start,TimeValue_Final;     struct timezone TimeZone_Start,TimeZone_Final;

    //Sequential
    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    result = seq(A, n, element);
    gettimeofday(&TimeValue_Final, &TimeZone_Final);

    printf("Sequential Binary Search: %lf\n", timecalc(TimeValue_Start,TimeValue_Final));
    if (result == -1){   printf("\tElement not found\n");}    else{   printf("\tElement found at: %d\n", result);}

    //Parallel
    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    result = parallel(A, 0, n-1, element);
    gettimeofday(&TimeValue_Final, &TimeZone_Final);

    printf(" Binary Search: %lf\n", timecalc(TimeValue_Start,TimeValue_Final));
    if (result == -1){   printf("\tElement not found\n");}    else{   printf("\tElement found at: %d\n", result);}
}
```