# NAME – GAUTAM CHANDRA SAHA REG NO – 201900099 DATE – 02/02/2022

**Q1. Write a menu driven Open-MP program in C to implement the following clauses:**

**a) PRIVATE b) FIRSTPRIVATE c) LASTPRIVATE d) ORDERED**

#include <omp.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

//each thread writes all numbers from 0 to MAX\_THREAD

void private\_clause(){

printf("\nNow running PRIVATE clause\n");

int a, i, thread\_id;

#pragma omp parallel for private(a, thread\_id)

for (i = 0; i < omp\_get\_max\_threads(); i++){

for (a = 0; a < omp\_get\_max\_threads(); a++){

thread\_id = omp\_get\_thread\_num();

printf("Thread #%d says \"%d\"\n", thread\_id, a);

}

}

return;

}

//each thread writes a number between 0 to MAX\_THREAD //all threads collectively write all such numbers unquely

void first\_private(){

printf("\nNow running FIRSTPRIVATE clause\n");

int i, j, r, thread\_id;

printf("Enter the value till which the threads should loop: ");

scanf("%d", &r);

// #pragma omp parallel for firstprivate(r) private(j, thread\_id)

for (i = 0; i < omp\_get\_max\_threads(); i++){

for (j = 0; j <= r; j++){

thread\_id = omp\_get\_thread\_num();

printf("Thread #%d says \"%d\"\n", thread\_id, i);

}

}

return;

}

//each thread rolls a die

//the number rolled by the last thread is retained

void last\_private(){

printf("\nNow running LASTPRIVATE clause\n");

int a = 10, r, thread\_id, i;

#pragma omp parallel for firstprivate(a) lastprivate(r) private(thread\_id)

for (i = 0; i < omp\_get\_max\_threads(); i++){

thread\_id = omp\_get\_thread\_num();

srand(thread\_id);

r = rand() % a + 1;

printf("Thread #%d rolled a die and got a \"%d\"\n", thread\_id, r);

}

printf("\nThe value of the die roll is now %d\n", r);

return;

}

//threads iterate over a range, treating it as a serial construct

void ordered(){

printf("\nNow running ORDERED clause\n");

int i, thread\_id;

#pragma omp parallel for ordered private(thread\_id)

for (i = 0; i <= 10; i++){

thread\_id = omp\_get\_thread\_num();

#pragma omp ordered

printf("Thread #%d says \"%d\"\n", thread\_id, i);

}

return;

}

//the menu

int menu(void){

printf("(1) - Run the implementation of the PRIVATE clause\n");

printf("(2) - Run the implementation of the FIRSTPRIVATE clause\n");

printf("(3) - Run the implementation of the LASTPRIVATE clause\n");

printf("(4) - Run the implementation of the ORDERED clause\n");

printf("(0) - Exit the program\n");

printf("Enter your choice: ");

int choice = 0;

scanf("%d", &choice);

switch (choice){

case 0:

break;

case 1:

private\_clause();

break;

case 2:

first\_private();

break;

case 3:

last\_private();

break;

case 4:

ordered();

break;

default:

printf("\nInvalid choice");

choice = 5;

break;

}

return choice;

}

int main(void){

while (menu()){

}

return 0;

}

**OUTPUT**

Text

Description automatically generated

Text

Description automatically generated

**Q2. Run a parallel prog to display/print A for 1 Lakh times using different cores.**

#include <stdio.h>

#include <omp.h>

int main(){

double sum = 0;

#pragma omp parallel

{

int i = 0;

while (i < 4){

double start, end;

start = omp\_get\_wtime();

if (omp\_get\_thread\_num() == i){

for (i = 0; i < 25000; i++)

printf(" A ");

}

end = omp\_get\_wtime();

printf("\nFunction used in Core : %d ", omp\_get\_thread\_num());

printf("\nTime taken by Core %d is : %f ", omp\_get\_thread\_num(), end - start);

sum += (end - start);

i++;

}

}

printf("\n Total execution time : %f ", sum);

return 0;

}

**OUTPUT**

Background pattern

Description automatically generated

**Q3. How to calculate execution time of a C program without omp.h.**

#include <stdio.h>

#include <time.h>

int main(){

clock\_t start, end;

start = clock();

for (int i = 0; i < 100000; i++)

printf("A ");

end = clock();

double time\_taken = ((double)end - start) / CLOCKS\_PER\_SEC; // in seconds

printf("\nIt took %f seconds to execute \n", time\_taken);

return 0;

}

**OUTPUT**

Background pattern

Description automatically generated