Q:What is DE-REENTRANT? Why we are using it in compiling a threaded program?

ANS: DE-REENTRANT tells the compiler to use the declarations. Defining DE-REENTRANT causes the compiler to use thread safe version of several functions in the C library.

Q:Create an array of 1000,also create 10 threads where each thread adds 100?

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int chunk\_size=100;

int array[1000];

void \* Add(void \*arg)

{

int start;

int end = start + chunk\_size;

int sum=0;

for(i=start;i<end;i++)

{

sum = sum +array[i];

}

return ((void\*)sum);

}

int main(void)

{

Int sum=0;

void status\_t1,status\_t2,status\_t3,status\_t4,status\_t5,status\_t6,status\_t7,status\_t8,status\_t9,status\_t10; pthread\_tthread\_t1,thread\_t2,thread\_t3,thread\_t4,thread\_t5,thread\_t6,thread\_t7,thread\_t8,thread\_t9,thread\_t10;

pthread\_create(&thread\_t1,NULL,Add,(void\*)(0\*chunk\_size));

pthread\_create(&thread\_t2,NULL,Add,(void\*)(1\*chunk\_size));

pthread\_create(&thread\_t3,NULL,Add,(void\*)(2\*chunk\_size));

pthread\_create(&thread\_t4,NULL,Add,(void\*)(3\*chunk\_size));

pthread\_create(&thread\_t5,NULL,Add,(void\*)(4\*chunk\_size));

pthread\_create(&thread\_t6,NULL,Add,(void\*)(5\*chunk\_size));

pthread\_create(&thread\_t7,NULL,Add,(void\*)(6\*chunk\_size));

pthread\_create(&thread\_t8,NULL,Add,(void\*)(7\*chunk\_size));

pthread\_create(&thread\_t9,NULL,Add,(void\*)(8\*chunk\_size));

pthread\_create(&thread\_t10,NULL,Add,(void\*)(9\*chunk\_size));

pthread\_join(thread\_t1,(void\*\*) & status\_t1);

pthread\_join(thread\_t2,(void\*\*) & status\_t2);

pthread\_join(thread\_t3,(void\*\*) & status\_t3);

pthread\_join(thread\_t4,(void\*\*) & status\_t4);

pthread\_join(thread\_t5,(void\*\*) & status\_t5);

pthread\_join(thread\_t6,(void\*\*) & status\_t6);

pthread\_join(thread\_t7,(void\*\*) & status\_t7);

pthread\_join(thread\_t8,(void\*\*) & status\_t8);

pthread\_join(thread\_t9,(void\*\*) & status\_t9);

pthread\_join(thread\_t10,(void\*\*) & status\_t10);

sum=sum+ status\_t1 + status\_t2 + status\_t3 + status\_t4 + status\_t5 + status\_t6 + status\_t7 + status\_t8 + status\_t9 + status\_t10 ;

printf("\nsum is %d\n", sum);

return 0;

}