1.Explain clearly whether the following program using OpenMP directives will compile correctly. Correct the program if it does not compile correctly. What are the possible outputs from this program? Explain your answer clearly.

```
#include <omp.h>
#include <stdio.h>
int main() {
int section_count = 0;
#pragma omp parallel
#pragma omp sections firstprivate(section_count)
{
#pragma omp section
{
section_count++;
printf( "section_count= %d\n", section_count );
}
#pragma omp section
{
section_count++;
printf( "section_count= %d\n", section_count );
}
}
return 0;
}
```

2.(a)Consider the following code. What are the entries of the array A after the program terminates? Parallelize the for loop using OpenMP directives. Explain why this OpenMP code will be inefficient.

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

```
int main()
{
int i,j, A[5];
j=4;
for(i=0;i<5;i++)
{
j=j*2;
A[i]=j;
}
}
```

3.Change the program in part (a) so that the parallelization of the for loop using OpenMP is more efficient. You can assume any extra function you like, but you need to explain it clearly.

Explain clearly whether the following program using OpenMP directives will compile correctly. Write a corrected version of the program if the program does not compile correctly. What does the program do?

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define N 50
#define CHUNKSIZE 5
int main (int argc, char *argv[])
{
int i, chunk, tid;
float a[N], b[N], c[N];
for (i=0; i < N; i++)
a[i] = b[i] = i * 1.0;
chunk = CHUNKSIZE;
#pragma omp parallel for \
shared(a,b,c,chunk) \
private(i,tid) \
schedule(static,chunk)
{
tid = omp_get_thread_num();
for (i=0; i < N; i++)
{
c[i] = a[i] + b[i];
printf("tid= %d i= %d c[i]= %f\n", tid, i, c[i]);
}
}
}
```

4.
```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main(void){
int i;
int x;
x=44;
#pragma omp parallel for lastprivate(x)
for(i=0;i<=10;i++){
x=i;
printf("Thread number: %d x: %d\n",omp_get_thread_num(),x);
}
printf("x is %d\n", x);
}
```

A partial output from this program is given below. Complete the remaining part of the output and explain your answer.

```
Thread number: 7 x: 10
Thread number: 3 x: 6
Thread number: 2 x: 4
Thread number: 2 x: 5
```