1. (a) Explain the advantages and disadvantages of arrays, linked lists and hash tables.

   (b) Give pseudo code for binary search on a sorted array of integers.

   (c) Explain $O(f(n))$. Say why binary search is $O(\log_2(n))$.

2. (a) Explain the two's complement format for storing signed integers in binary.

   (b) Give a brief overview of the IEEE 754 floating point standard.

   (c) Show how the number $-\frac{1}{3}$ would be stored as a `float`. (A normalised float is written as $\pm 2^{e-127}(1.s)$ with 8 bits for e and 23 bits for s).

3. (a) What is a SMP system? Give two advantages and two disadvantages of this type of system when compared with a network of workstations.

   (b) Give a description of direct-mapped and fully-associative caches. Why are all caches not fully-associative?

   (c) Figure 1 shows the graph of the performance of a 4 processor SMP machine, calculating large dot products of various sizes. The tests were done first running one process at a time and then running 4 processes at a time and plotting the performance of one of them. What can you say about the caches and memory system of the computer in question?

4. (a) Figure 2 shows some rather dubious C code. List 3 improvements which could be made to this code.

   (b) A user enters the file name:

   veryveryveryvery. . . veryveryveryverylongfilename

   and the program runs normally but crashes when it finishes. The debugger shows it crashed trying to run nonexistent code at address `0x76657279`. Can you explain what happened in terms of memory allocation and the stack? (Hint: The ASCII for 'e' is `0x65`).

   (c) Describe three common optimisations compilers perform. Give example piece of code showing where each might be applied.

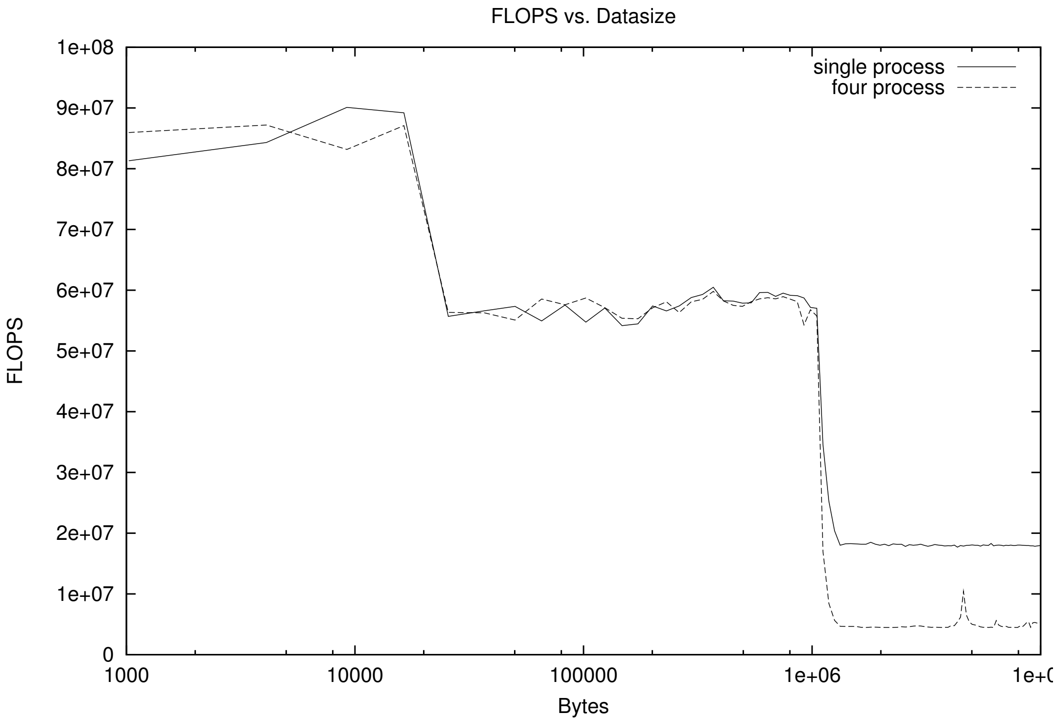

Figure 1: Performance on Large Dot Products

```
#define DIM 3

FILE *fp;

int main() {
    char filename[100];
    int i, number;
    float *list;

    printf("Data file: ");
    gets(filename);

    fp = fopen(filename, "r");
    fscanf(fp, "%d", &number);

    list = malloc(4*DIM*number);
    for( i = 0; i < number; i++ ) /* Loop around */
        read_xyz(&(list[DIM*i]));

    free(list);
    return 0;
}

void read_xyz(float *r) {
    int i;

    for( i = 0; i <= DIM; i++ )
        fscanf("%lf", &(r[i]));
}
```

Figure 2: Dubious C code