

EXAMINATIONS — 2008

FINAL EXAM

SWEN 201/COMP 206
PROGRAM AND
DATA STRUCTURES

Time Allowed: 3 hours

Instructions:

- Attempt all questions.
- Write your student ID number at the top of each sheet.
- There are 180 possible marks on the exam.
- Make sure your answers are clear and to the point.
- Non-programmable calculators without full alphabetic keys are permitted.
- Non-electronic foreign language dictionaries are permitted.
- Refer to the Appendix.
- No other reference material is allowed.
- Answer in the appropriate heavily outlined boxes or follow the instructions given in the questions.

Question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
Total	

Student ID:

Student ID:

Question 1. C Basics

[14 marks]

(a) [2 marks] In the box below state the output of the following program.

The program uses `strcpy`, a function defined in `string.h`.

`strcpy(dst, src)` copies the string `src` to `dst` (including the terminating `'\0'` character).

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

#define SIZE 6

int main() {
    int i;
    char m[SIZE];

    strcpy(m, "ABC12");

    for(i=0 ; i<SIZE-1; i++) {
        printf("%d=%c\n", i, m[i]+2);
    }
    return 0;
}
```

(Question 1 continued on next page)

Student ID:

(Question 1 continued)

(b) [12 marks] The C program below is supposed to read two integers from the terminal and print the list of integers from the lower to the higher.

```
#include <stdio.h>

int swap(int* x, int *y);

int main() {
    int i, x, y;

    printf("Value A?");
    scanf("%s", x);
    printf("Value B?");
    scanf("%s", y);

    if(x>y)swap(x, y);

    for(i=x, i<y, i++){
        printf("%d ", i)
    }
    printf("\n", i);
}

/* swaps the values of the two arguments */
int swap(int* x, int *y){
    int tmp;
    tmp=*y;
    *x=*y;
    *y=tmp;
}
```

It should behave like this:

```
% ./a.out
Value A? 4
Value B? 9
4 5 6 7 8 9
% ./a.out
Value A? 8
Value B? 5
5 6 7 8
```

(Question 1 continued on next page)

Student ID:

(Question 1 continued)

Unfortunately this code contains several errors. In the box below identify the errors and provide corrections:

Student ID:

Question 2. Dynamic Data Structures

[22 marks]

You have been employed as a C programmer on a corpus linguistics research project. Corpus linguists investigate language by studying the properties of a large body (a corpus) of text, almost always using a computer.

One statistic in which you are interested is how frequently each word in a text occurs. You decide to write a C program which will read words from a text file and will record how frequently each word occurs. You decide to use a binary search tree to record the data, where each node records a word and its frequency. You need functions to:

- print out a tree (showing the words in alphabetical order),
- insert a single word into a tree,
- read a file and insert all the words it contains into a tree, and
- open a file.

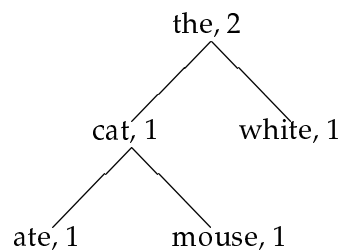
The method to insert a word to the tree can be described as follows:

- if the current node is NULL then add a new node containing the word with frequency 1;
- if the word is the same as the one at the current node, increment the frequency;
- if the word is less than the one at the current node, insert the data in the left subtree;
- if the word is more than the one at the current node, insert the data in the right subtree;

If the input file contains:

```
the cat ate the white mouse
```

then the following tree will be created:



(Question 2 continued on next page)

Student ID:

(Question 2 continued)

The header file for your program looks like this:

```
#define WORDSIZE 20
#define node_size sizeof(node)

struct treeitem {
    char word[WORDSIZE];
    int freq;
    struct treeitem* left;
    struct treeitem* right;
};

typedef struct treeitem node;

node* insert(node* tree, char* w);

void printtree(node* tree);

node* read_from_file(FILE* fp, node* tree);

FILE* openfile(FILE* fp, char* fname);
```

The main function looks like this:

```
int main(int argc, char* argv[])
{
    FILE *fp1;
    node* the_tree;

    fp1 = openfile(fp1, argv[1]);          /* Open the file. */
    the_tree=read_from_file(fp1, NULL); /* Build the tree. */
    printf("%s %s\n", "Frequency", "word");
    printtree(the_tree);                  /* Print the tree. */
    return fclose(fp1);
}
```

Student ID:

(Question 2 continued)

(a) [8 marks] In the box below implement the `printtree` function, which will print the words out in alphabetical order. Given the tree above, this function will produce:

```
1 ate
1 cat
1 mouse
2 the
1 white
```

```
void printtree(node* tree){
}

```


Student ID:

Question 3. C and C++

[18 marks]

(a) [6 marks] In the box below state the syntax of C's `struct` and `union` constructs and describe a situation when a `struct` would be used and a situation when a `union` would be used.

(Question 3 continued on next page)

Student ID:

(Question 3 continued)

(b) [6 marks] In the box below explain the two parameter passing mechanisms provided by C++.

Student ID:

(Question 3 continued)

(c) [3 marks] C provides `malloc` and `free` for memory management. Explain what these functions do.

(d) [3 marks] C++ provides additional constructs for memory management. State what these are and what advantages they have over the constructs offered by C.

Student ID:

Student ID:

Question 4. C++ programming

[30 marks]

You have been hired as a C++ programmer by a company which operates multi-storey car parks. They are not making as much profit as they would like, and they want to run some simulations to help them understand their business better.

You are just beginning to design the simulation, so you adopt a very simple model of a parking building. The information you need to record for a parking building are:

- its capacity;
- the number of cars it contains (never greater than its capacity);
- the income which the building has earned (it costs \$1.00 to park a car in a building).

No vehicle can enter a building which is full, and, of course, no vehicles can leave an empty building.

You decide to use a C++ class, `Parking`, to represent a parking building. You realise that you can model the arrival a car at the building using the `++` operator and the exit of a car using the `--` operator.

You also decide to implement `>>` and `<<` operators for parking buildings. The `>>` operator will prompt the user for a capacity for the parking building, the `<<` operator will report the current state of the building.

For example, given the following main function:

```
int main(){
    int i;
    Parking p1;

    cin >> p1; /* Get a capacity for p1 */

    Parking p2 = p1; /* p2 is just like p1 */

    for(i=0; i<200; i++){p1++;} /* 200 attempted arrivals at p1 */
    for(i=0; i<100; i++){p2++;} /* 100 attempted arrivals at p2 */

    cout << "Building p1 -- " << p1 << endl;
    cout << "Building p2 -- " << p2 << endl;

    for(i=0; i<100; i++){p1--;} /* 100 attempted exits from p1 */
    for(i=0; i<200; i++){p2--;} /* 200 attempted exits from p2 */

    cout << "Building p1 -- " << p1 << endl;
    cout << "Building p2 -- " << p2 << endl;
}
```

(Question 4 continued on next page)

Student ID:

(Question 4 continued)

you should get the following behaviour:

Capacity? 150

Building p1 -- Current = 150 Capacity = 150 Income = \$150

Building p2 -- Current = 100 Capacity = 150 Income = \$100

Building p1 -- Current = 50 Capacity = 150 Income = \$150

Building p2 -- Current = 0 Capacity = 150 Income = \$100

(a) [5 marks] In the box below define a suitable C++ class Parking:

Student ID:

(Question 4 continued)

(b) [5 marks] In the box give C++ code (including comments) for the assignment operator for Parking:

(c) [5 marks] In the box give C++ code (including comments) for the post-increment operator for Parking:

(Question 4 continued on next page)

Student ID:

(Question 4 continued)

(d) [5 marks] In the box give C++ code (including comments) for the post-decrement operator for Parking:

(e) [5 marks] In the box give C++ code (including comments) for the << operator for Parking:

(Question 4 continued on next page)

Student ID:

(Question 4 continued)

(f) [5 marks] In the box give C++ code (including comments) for the >> operator for Parking:

Student ID:

Question 5.

[18 marks]

C++ provides support for *polymorphism* in several ways. In the box below:

- state what forms of polymorphism C++ supports,
- explain how they are supported, and
- explain what advantages polymorphism gives to the programmer.

Student ID:

Question 6. File Structure Fundamentals

[23 marks]

(a) [6 marks] Explain the difference between primary and secondary file organisation.

(b) [17 marks] A file contains 20,000 student records of fixed length. Each record has the following fields: StudentID (length = 9 bytes), Name (length = 30 bytes), Address (length = 40 bytes), DoB (length = 8 bytes). Assume that the file blocks are stored contiguously. The block size for the file is $B = 512$ bytes.

(i) [4 marks] Calculate the record size L in bytes. Show your working.

(ii) [4 marks] Calculate the blocking factor f and the number of file blocks b . Assume an unspanned file organisation. Show your working.

Student ID:

(iii) [4 marks] Calculate the average number of block accesses needed to perform a linear search for a random record in the file given its StudentID. Show your working.

(iv) [5 marks] Calculate the average number of block accesses needed to perform a binary search for a random record in the file given its StudentID. Assume the file is ordered by StudentID. Show your working.

Student ID:

Question 7. B -Trees and B^+ -Trees

[30 marks]

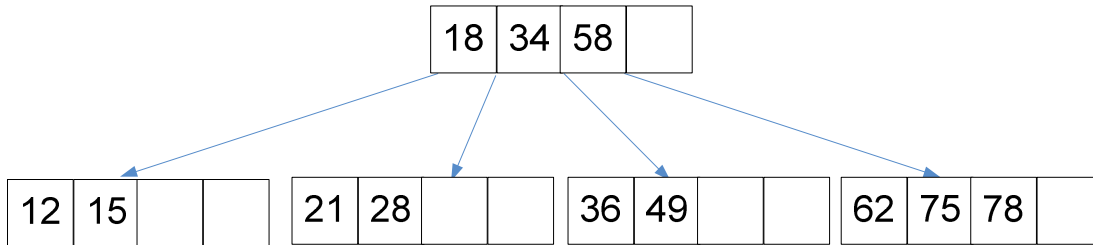
(a) [10 marks] State the definition of the index structure B -tree with order $p = 2m + 1$ and height h . Indicate which rules in the definition of a B -tree keep it well-balanced.

Student ID:

(Question 7 continued)

(b) [10 marks]

Consider the *B*-tree of the order 5 illustrated below.



Update the *B*-tree by deleting the key values 58, 75, 21 (in this order). In your answer, show the *B*-tree after each deletion and briefly describe what you have done.

The *B*-tree after deleting key value 58:

The *B*-tree after deleting key value 75:

Student ID:

The *B*-tree after deleting key value 21:

(c) [5 marks]

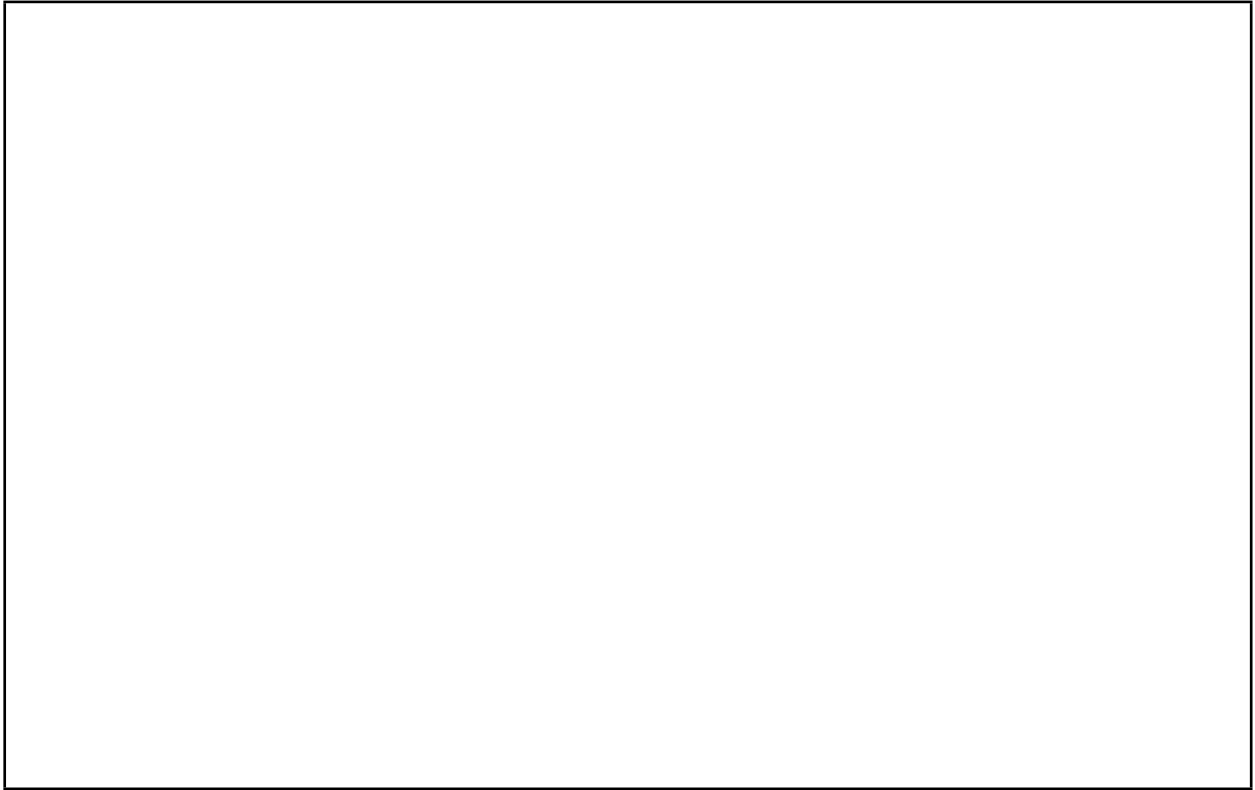
Consider a *B*-tree file of order $p = 2m + 1 = 99$ that contains $r = 10^6$ records. How many disk accesses will it take to retrieve a random record from the file using the *B*-tree in the worst case?

Show your working.

Student ID:

(Question 7 continued)

(d) [5 marks] Explain the difference between B -tree and B^+ -tree index structures.



Student ID:

Question 8. Hashing

[25 marks]

(a) [6 marks] Explain the following basic terms:

- What is a static hash file?


- What is an overflow record?

- What is linear probing?

Student ID:

(Question 8 continued)


(b) [9 marks] Describe the common characteristics of dynamic hash file schemes that allow a hash file to expand and shrink dynamically.



Student ID:

(Question 8 continued)

(c) [10 marks] Discuss the advantages and disadvantages of using (a) a heap file, (b) a sequential file, and (c) an extendible hash file. Explain in particular which file operations (insertion, deletion, search) can be performed efficiently in each of the cases, and which are expensive.



Appendix

File Performance Formulae

- blocking factor $f = \lfloor \frac{B}{L} \rfloor$
- number of blocks $b = \lceil \frac{r}{f} \rceil$
- external sort-merge $N = 2b \cdot (1 + \lceil (\log_{n-1} b) - 1 \rceil)$
- number of buffers n

B-tree (the worst case)

- height $h = 1 + \lfloor \log_{m+1} \frac{r+1}{2} \rfloor$
- number of leaves N_{leaves} : $2(m+1)^{h-2} \leq N_{leaves} \leq (2m+1)^{h-1}$

B⁺-tree (the worst case)

- height $h = 2 + \lfloor \log_{m+1} \frac{r}{2m} \rfloor$
- number of leaves $N_{leaves} = \lceil \frac{r}{m} \rceil$

Index-Sequential File with a B-tree

- number of sequence sets s : $\lceil \frac{r}{f} \rceil \leq s \leq \lceil \frac{2r}{f} \rceil$
