UNIVERSITY of GUELPH

CIS2520 Data Structures

Fall 2011, Midterm

All answers must be justified in a clear, concise and complete manner. There are 12 questions, which are worth 47 marks in total. You do not have to answer all the questions, since 40 marks are enough to guarantee you 100% on this test.

1) 5 MARKS

Consider four functions f_1, f_2, g_1 and g_2 from $\mathbb{Z}_+$ to $\mathbb{R}_+$, where $\mathbb{Z}_+$ is the set of positive integers and $\mathbb{R}_+$ is the set of positive real numbers. Assume each function is defined on a neighbourhood of infinity.

(a) Show that if f_1 is $O(g_1)$ and f_2 is $O(g_2)$ then f_1f_2 is $O(g_1g_2)$.
(b) Show that if f_1 is $O(g_1)$ and f_2 is $O(g_2)$ then f_1/f_2 is not necessarily $O(g_1/g_2)$.

2) 5 MARKS

Consider the function $f : \mathbb{Z}_+ \to \mathbb{R}_+$

$n \mapsto 2n^4+9n^3\sqrt{n}-17$

where $\mathbb{Z}_+$ is the set of positive integers and $\mathbb{R}_+$ is the set of positive real numbers.

(a) Show that f is defined on a neighbourhood of infinity.
(b) Show that f(n) is $O(n^4)$.

3) 3 MARKS

Consider the algorithm below. Let n be the input size (i.e., the matrices are of order n; they are $n \times n$ matrices). Assume the slowest primitive operation consumes *tmax* time.

(a) What is the worst-case number of primitive operations executed by the algorithm?
(b) What is the worst-case running time?
(c) What is the time complexity of this algorithm, expressed using O-notation?

```
0           function SumMatrices (A, B)
1               for i=0 to A.order−1
2                   for j=0 to A.order−1
3                       S[i][j]=A[i][j]+B[i][j]
4               return S
```

4) 5 MARKS

Let u and v be two integers. Assume 0≤u≤v. The sum of all integers from u to v is:

$$\sum_{k=u}^{v} k = u+(u+1)+\dots+v = [v(v+1)-u(u-1)]/2$$

Same questions as in **3)** for the algorithm below.

```
0           function TransposeMatrix (A)
1               for i=0 to A.order−2
2                   for j=i+1 to A.order−1
3                       exchange A[i][j] with A[j][i]
```

5) 3 MARKS

Consider a sorted integer array **A** and three integers **v**, **first** and **last**.
Write in pseudocode a recursive binary search algorithm which returns a value **k** such that **A[k]=v** and **first≤k≤last**, or **−1** if no such **k** can be found.

6) 4 MARKS

You are a client of the *Student* library as described in Assignment 1:

```
extern void InitializeStudent (char *name, int grade, Student *S);
extern char *NameOfStudent (Student S);
extern int GradeOfStudent (Student S);
extern void FreeStudent (Student *S);
```

You are implementing a *List of Students* library, and this is what you got so far:

```
typedef struct {
      Student items[100];
      int size;
} List;

void Initialize (List *L) {L−>size=0;}
void Insert (Student X, int position, List *L) {
      int i;
      for (i=L->size; i>position; i--)
           L->items[i]=L->items[i-1];
      L->items[position]=X;
      L->size++;
}

void ShowItem (int position, List *L) {
      Student S=L->items[position];
      printf("%s %d\n",NameOfStudent(S),GradeOfStudent(S));
}
```

You have also written the following program to test your implementation:

```
int main(void) {
      Student S;
      List L;
      Initialize(&L);
      InitializeStudent("John",75,&S);
      Insert(S,0,&L);
      FreeStudent(&S);
      InitializeStudent("Mary",95,&S);
      Insert(S,0,&L);
      FreeStudent(&S);
      ShowItem(0,&L);
      ShowItem(1,&L);
}
```

The program compiles fine, but the output is

```
Mary 95
Mary 75
```

instead of

```
Mary 95
John 75
```

(a) Explain why, and **(b)** modify `Insert` to fix the problem.

7) 2 MARKS

A linked list does not necessarily come with a header node.
Consider the following concrete data structure definition for lists of integers:

```
typedef struct ListTag {
     int item;
     struct ListTag *next;
} List;
```

Write a recursive function that outputs the items of a list in the backward direction (i.e., from the last to the first). Note that you are not asked to reverse the list.

8) 2 MARKS

Consider the following concrete data structure definition for queues of integers:

```
typedef struct {
    int items[100];
    int size;
    int head;
} Queue;
```

Write the function `Dequeue`, knowing that queues are implemented using circular arrays.

9) 5 MARKS

When a structure is passed to a function, it is actually a copy of the structure that is passed to the function. If the structure is large, it is more efficient to pass a pointer to the structure instead. Moreover, note that a linked list does not necessarily come with a header node.

Consider the following concrete data structure definition
for stacks of items of type `SomeLargeStructure`:

```
typedef struct StackTag {
    SomeLargeStructure item;
    struct StackTag *next;
} Stack;
```

(a) Write a recursive function `Size` to calculate the number of items in a stack.
(b) Write a function `Push`. If ever you need to manipulate items of type `SomeLargeStructure` (e.g., compare two such items), you may assume that a function is available to perform the task.

10) 4 MARKS

Given the Stack and Queue operations below,
describe in pseudocode an algorithm for reversing a stack using a queue.

```
SCreate: ∅ → Stack[T]
SPush: TxStack[T] → Stack[T]
SPop: Stack[T] → Stack[T]
SFull: Stack[T] → Boolean
```

```
SEmpty: Stack[T] → Boolean
SSize: Stack[T] → N
STop: Stack[T] → T

QCreate: ∅ → Queue[T]
QEnqueue: TxQueue[T] → Queue[T]
QDequeue: Queue[T] → Queue[T]
QFull: Queue[T] → Boolean
QEmpty: Queue[T] → Boolean
QSize: Queue[T] → N
QHead: Queue[T] → T
```

11) 6 MARKS

Let **T** be a nonempty set and let **N** be the set of nonnegative integers. A ***priority queue*** of items of type **T** is a tuple $\mathbf{((I_1,P_1), (I_2,P_2), \ldots, (I_n,P_n))}$, where each $\mathbf{I_i}$ belongs to **T** and each $\mathbf{P_i}$ to **N**, such that: $\forall \mathbf{i} \in \mathbf{1..n-1}, \mathbf{P_i \leq P_{i+1}}$. We say that the item $\mathbf{I_i}$ is of priority $\mathbf{P_i}$.

The operations defined on priority queues are: the constructor **Create** (creates an empty priority queue); the mutators **Insert** (inserts an item **I** of priority **P** into a priority queue **Q**) and **Remove** (removes from **Q** an item of highest priority); the accessors **Full** (determines whether **Q** is full), **Empty** (determines whether **Q** is empty), **Size** (finds the number of items in **Q**), **Priority** (finds the highest priority in **Q**), and **Item** (finds an item of highest priority).

(a) Let **PQueue[T]** be the set of all priority queues of items of type **T**, and let **Boolean** be the set of Boolean values. Specify the domain and codomain of the operations **Create**, **Insert**, **Remove**, **Full**, **Empty**, **Size**, **Priority** and **Item**.

(b) Write a postcondition for **Insert(I,P,Q).** This postcondition should involve: the accessors **Empty**, **Size**, **Priority** and **Item**; the keyword **old**; the logical negation ¬, conjunction ∧, and implication →.

(c) Write an axiom that involves the operations **Insert**, **Remove**, and **Priority**.

12) 3 MARKS

Explain the aim and use of the preprocessor macro `assert()`.