

Midterm Examination 1

ECE-242 Algorithms and Data Structures
Spring Semester, 2003

Wednesday, February 12, 2003

Print your name and ID number neatly in the space provided below; print your name at the upper right corner of every page. Place a picture ID on your table for verification.

Your Name:
ID Number:
Instructor Name:

This booklet should be six pages long including this page. If not, report this to the instructor or the TA. Do all problems in this booklet. Try not to spend too much time on any one question. Raise your hand if you have a question.

This is a closed book exam except for a single-sided half-page aid sheet. Foreign students can use a dictionary. Nothing else is allowed. Use terminology from the lecture notes. You must define any different terms before you use them. **Do all work in the space provided.** Ask the proctor if you need more paper. **Nothing on the back of the sheets will be graded.**

Question	Points	Score	Grader
Multiple Choice	15		
Short Answer 1	20		
Short Answer 2	11		
Short Answer 3	14		
Total	60		

(Multiple Choice, 15 points) Each question worths 3 points and it has only one correct answer. If we cannot understand what you marked you will receive no credit.

1) What is the output of the following `printf()` statement?

```
int n, m=6;

n=m++;
m+= --n;
printf("Your result is %d", m);
```

- (a) 11 (b) 12 (c) 13 (d) 14

2) What should a header `.h` file contain?

- (a) Function declarations (c) Preprocessor defines
(b) Function definitions (d) a) and c)

3) How many bytes are in a `integer`?

- (a) 2 (b) 4 (c) 8 (d) depends on compiler/architecture

4) What does the following function return?

```
int check_if_equal(float value)
{
    return(value=6.0);
}
```

- (a) 6 (b) 6.0 (c) 1 (d) not enough information to answer

5) Which of the following Big-Oh asymptotic complexity best describes $n^m \log_2 n + 3n^m + n$?

- (a) $O(n^{m+1})$ (c) $O(n^m)$
(b) $O(n^m + n)$ (d) $O(n)$

Short Answer, 45 points Read each question carefully. Be clear and careful when you write code. Answer in the specified areas only. **Marks will be deducted if you do not follow the instructions!** Credit for each question is shown next to the question.

[1. **Programming, 8+6+6 points**] For this problem, every program you write should be approximately 10 to 14 lines. **You will lose marks if your program is long!**

A *singly-linked list* is used to implement a *stack*. The first element of the linked list is the top of the stack. Therefore, `*head` points to the top of stack (or NULL when the stack is empty). The second element of the linked list is the one next to the top in the stack etc. The last record (stack bottom) always points to NULL. You are given the data structure and code for `Push()`.

```
typedef struct stack {
    char data;
    struct stack *next;
} node;

char Push(node **head, char data) {
    node *temp;
    temp = (node *) malloc(sizeof(node));
    temp->data = data;
    temp->next = *head;
    *head = temp;
return(data);
}
```

(a) Write code for the `Pop()` function.

```
/* Function pops, returns the top item of the stack, updates
   stack status and prints error message when stack is empty */

node *Pop(node **head) {

}

}
```

b) Write a **recursive** function `CopyStack()` that returns a duplicate copy of the stack. The original stack (linked-list) should not be modified or destroyed. *Do not* use loops.

```
/* creates a new copy of the stack and returns pointer to the top of new
stack */
node *CopyStack(node *head) {
```

```
}
```

(c) Write a **recursive** function `PrintStackBackwards()` that prints out the elements of the stack (linked-list) in reverse order (bottom character first, next to bottom next etc). *Do not* use loops or modify the original data structure.

```
void PrintStackBackwards(node *head) {
```

```
}
```

[2. Debugging, 11 points]

Circle the 7 to 10 *compiler errors (bugs)* and *run time errors* in the C program below. Next to each circle write a number (footnote). Reference this number at the end of the page with a 1-2 line explanation *whether* and *why* you believe it is a bug or a run-time error. *Negative marks* will be accounted for incorrect answers.

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

int main(int argc, char *argv[])
{
    int *ip, **myp, i, n;

    printf("\n How many integers to clear?");
    scanf("%d", &m);

    ip = (int *) malloc(4*n); /* allocate appropriate space /
    myp = &ip;

    for(i=0; i<=n; ++i)
    {
        printf("\n Value in allocated memory is %d", ip[i]);
        ip[i]=0;
    }
    free(ip);

    printf("\n The magic number is %d", **myp);

    return(1);
}
```

[3. Complexity, 7+7 points] Consider the following C program fragments. Line numbers at the left of the code are not part of the code but you can reference them in your answers.

(a) What is a tight Big-Oh time complexity bound for this code fragment? Explain in 2-3 lines.

```
1.      int i,j, n, m, k;
2.      for(i=0; i < n; i++)
3.          if (i == m)
4.              for(j=0; j<m; j++)
5.                  k=i;
```

$T(n, m) = O(\quad)$

(b) What is a tight Big-Oh time complexity bound for this code fragment? Explain in 2-3 lines.

```
1. int stranger(int n) {
2.     if (n < 1)
3.         return 1;
4.     else {
5.         if (n % 2 == 0)           /* n is even */
6.             return stranger(n-1) + stranger(n-1);
7.         else                       /* n is odd */
8.             return stranger(n-2) + 1;
9.     }
10. }
```

$T(n) = O(\quad)$