Name: _____

CS 24
Midterm Exam
Summer 2013 C

Data and Memory	/16
FILE I/O	/16
List ADT	/16
Other	/12
Total	**/60**

Please do not begin until told to do so.

Exam Rules

1. No one may leave for any reason and come back to work on the test. If you need to use the bathroom, do it now.
2. There may not be any extra material on your desk or adjacent ones.
3. You may not wear any sunglasses, hats, or hoods.
4. You may not use any electronic devices of any kind. Any headphones need to be put away.
5. Please don't sit in the front row of the classroom.
6. If you appear to be looking around, you will be moved to the front row.

Use for Reference throughout the Test

```
struct BasicData {
      int _key;
      char _buf[8];
};

struct BasicNode {
      struct BasicData _data;
      struct BasicNode *_a;
      struct BasicNode *_b;
};


void do_something_interesting(int n) {
      struct BasicNode nodes[4];
      struct BasicNode *tmp;
      if (n < 4)
            tmp = nodes;
      else
            tmp = malloc(sizeof(struct BasicNode) * n);
      tmp[0]._a = NULL;
      tmp[n - 1]._b = NULL;
      for (int i = 0; i < n; ++i) {
            tmp[i]._data._key = i;
            if (i > 0)
                  tmp[i - 1]._b = tmp[i];
            tmp[i];
      }

      if (n >= 4)
            free(tmp);

}

int main() {
      for (int i = 0; i < 10; ++i) {
            do_something_interesting(i);
      }
      return 0;
}
```

Data and Memory

1. (6 pts.) What are the three segments of memory (as described in class) and what is contained within each?

2. (2 pts.) [From page 2] What is the size, in bytes, of the `BasicData` structure? Recall that a character is 1 byte, an integer 4 bytes, and pointers are 4 bytes.

3. (2 pts.) What is the size, in bytes, of the `BasicNode` structure?

4. (6 pts.) What are all the elements and their sizes that make up the simplified activation record (data only) of the function do_something_interesting? Hint: the sum of these sizes should be the total size of do_something_interesting's simplified activation record.

FILE I/O

File I/O Function Prototypes

- `FILE* fopen(char *filename, char *mode);`
- `int fclose(FILE *stream);`
- `int fgetc(FILE *stream);`
- `char *fgets(char *buf, int n, FILE *stream);`
- `int fputc(int c, FILE *stream);`
- `int fputs(char *buf, FILE *stream);`

5. (16 pts.) Complete the code to copy the contents of `src_file` to `dst_file` replacing any newlines with spaces. The program should output the number of newlines that were removed. That is, if the contents of the file `src` are "`Julie\nis\namazing!`" the contents of the file `dst` after the program executes should be "`Julie is amazing!`" and the program should produce output similar to "`Replaced 2 newlines`". Note that you need not perform any error checking.

```
#include <stdio.h>
int main() {
    FILE *src_file = fopen("src", "r");
    FILE *dst_file = fopen("dst", "w");









    return 0;
}
```

List ADT

6. (8 pts.) In project one you wrote the function `list_push_back` that adds an item to the end of the list. Complete the linked-list function implementation for `list_insert_at` that inserts items into the provided position in the list. Note that you can assume the value of position is valid. Position 0 indicates insert at the front of the list, and position equal to the size of the list indicates insert at the end of the list. Recall that the `List` structure has only the attribute `_head`, and the `Node` structure has the attributes `_data`, and `_next`.

```
int list_insert_at(struct List *list, int position,
                   char *item) {
    struct Node *node = malloc(sizeof(struct Node));
    if (!node)
        return 0;
    node->_data = item;
    if (position == 0) {  /* Insert at the front */
        node->_next = list->_head;
        list->_head = node;
    }
    else {









    }
    return 1;
}
```

7. (2 pts.) What is the big-O of the linked-list implementation of the `list_insert_at` function?

8. (6 pts.) Assume we optimized the linked-list implementation of the List ADT such that we could get the size of the list in O(1). Describe a workload, or usage of the List ADT, in which it would be beneficial to use the array-list implementation of the List ADT. Justify your answer.

Other

9. (4 pts.) What is the output of the following program when executed via:
`./a.out unwind their cassette slowly tomorrow evening`

```
#include <stdio.h>
int main(int argc, char *argv[]) {
  for(int i = 0; i < argc; ++i) {
    printf("%c", argv[i][2]);
  }
  printf("\n");
  return 0;
}
```

10. (2 pts.) [From page 2] What is the worst-case running time, big-O, of the `do_something_interesting` function?

11. (2 pts.) [From page 2] What is the big-O of the `main` function?

12. (4 pts.) [From page 2] What is the smallest number of test cases needed to perform complete branch-testing of the `do_something_interesting` function? What is one minimal set of values for parameter `n` that results in complete branch-testing? Hint: the number of items in the set should match your first answer.