

ECE 264 Exam 2

01:30-02:20PM, March 06, 2009

1 Structure and Function (4.8 points, outcome 2)

This set of questions asks you to define a structure called `Person` and implement the associated functions. You do not have to write the `main` function, nor do you need to include header files.

1.1 Structure (0.7 point)

Create a structure called `Person`. This structure has the following attributes:

- `int` called `age`,
- `char *` called `name`,
- `char *` called `address`.

1.2 Constructor (0.7 point)

Write the constructor for `Person`. The constructor

- takes three arguments in the same order as the attributes: `int`, `char *`, and `char *`,
- returns `Person *`,
- allocates memory as needed,
- assigns the arguments to the attributes.

1.3 Destructor (0.6 point)

Write the destructor for Person to release memory. The destructor takes one argument of Person * and returns nothing (void).

1.4 Copy (0.7 point)

Write the copy function for Person. This function

- takes one argument of Person *,
- creates a new Person * that has the same values in the attributes as the argument,
- returns the newly created object as Person *.

After the function returns, the newly created object has the same values in the attributes as the argument but the two objects do **not** share memory.

1.5 Assignment (0.7 point)

Write the assignment function. This function

- takes two arguments, one of Person ** and the other of Person *,
- checks whether the two arguments point to the same memory location. If they do, the function does nothing and returns.
- If they point to different memory locations, the function releases the memory occupied by the first argument,
- copies the second argument to the first argument,
- The function returns nothing (void).

After the function returns, the first argument has the same values in the attributes as the second argument but the two objects do **not** share memory.

1.6 Print (0.7 point)

Write a function that prints all attributes. The function takes one argument of `Person *` and returns nothing (`void`).

1.7 Objects (0.7 point)

Use the functions you have written to (1) create two `Person` objects.

- The first person is 20 years old. Her name is “Amy Jones” and she lives at “1020 First Street”.
- The second person is 21 years old. His name is “Tom Smith” and he lives at “9 Main Street”.

(2) print the two objects. (3) release the memory occupied by the two objects.

Answer:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct
{
    int age;
    char * name;
    char * address;
} Person;

Person * Person_construct(int a, char * n, char * ad)
{
    Person * s = malloc(sizeof(Person));
    s -> age = a;
    s -> name = malloc(strlen(n) + 1) * sizeof(char));
    strcpy(s -> name, n);
    s -> address = malloc(strlen(ad) + 1) * sizeof(char));
    strcpy(s -> address, ad);
    return s;
}
```

```

void Person_destruct(Person * s)
{
    free (s -> address);
    free (s -> name);
    free (s);
}

Person * Person_copy(Person * s)
{
    return Person_construct(s -> age, s -> name, s -> address);
}

void Person_assignment(Person ** p1, Person * p2)
{
    if ((*p1) == p2) { return; }
    Person_destruct (* p1);
    * p1 = Person_copy(p2);
}

void Person_print(Person * s)
{
    printf("Person: age=%d, name=%s, address=%s\n",
           s -> age, s -> name, s -> address);
}

int main(int argc, char * argv[])
{
    Person * p1 = Person_construct(20, "Amy Jones", "1020 First Street");
    Person * p2 = Person_construct(21, "Tom Smith", "9 Main Street");
    Person_print(p1);
    Person_print(p2);
    Person_destruct(p1);
    Person_destruct(p2);

    return 0;
}

```

2 Pointer (1.8 points)

What is the output of this program?

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

Name:

4

Seat:

```

void func1(char * p1, char * p2)
{
    if ((p1 == 0) || (p2 == 0)) { return; }
    char * temp = p1;
    p1 = p2;
    p2 = temp;
}

void func2(char ** p1, char ** p2)
{
    if ((p1 == 0) || (p2 == 0)) { return; }
    char * temp = * p1;
    * p1 = * p2;
    * p2 = temp;
}

void func3(char * p1, char * p2)
{
    char temp[60];
    if ((p1 == 0) || (p2 == 0)) { return; }
    strcpy(temp, p1);
    strcpy(p1, p2);
    strcpy(p2, temp);
}

char * findWord(char * str, int nword)
{
    char * charPtr = str;
    int wordCnt = 1;
    while ((charPtr != 0) && (wordCnt < nword))
    {
        charPtr = strchr(charPtr, ' ');
        if (charPtr == 0) { return 0; }
        charPtr++;
        wordCnt++;
    }
    return charPtr;
}

int main(int argc, char * argv[])
{
    char * str1;
    char * str2;
    char * str3;
    char * str4;

```

```
str1 = malloc(60 * sizeof(char));
str2 = malloc(60 * sizeof(char));
strcpy(str1, "ECE 264 W LAF IN");
strcpy(str2, "PU B12 Wed Fri exam");
func1(str1, str2);
printf("after func1\n\tstr1 = %s\n\tstr2 = %s\n\n", str1, str2);
func2(& str1, & str2);
printf("after func2\n\tstr1 = %s\n\tstr2 = %s\n\n", str1, str2);
str3 = findWord(str1, 3);
str4 = findWord(str2, 3);
func3(str3, str4);
printf("after func3\n\tstr1 = %s\n\tstr2 = %s\n\n", str1, str2);
free (str1);
free (str2);
return 0;
}
```

after func1

after func2

after func3

Answer:

```
after func1 /* calling func1 or not makes no difference */
    str1 = ECE 264 W LAF IN
    str2 = PU B12 Wed Fri exam
```

```
after func2
    str1 = PU B12 Wed Fri exam
```

Name:

6

Seat:

```

str2 = ECE 264 W LAF IN

after func3
    str1 = PU B12 W LAF IN
    str2 = ECE 264 Wed Fri exam

```

3 Complexity (0.4 point)

Derive the result of

$$\sum_{i=1}^n (2i - 1). \quad (1)$$

Your answer should use the symbol n .

Answer:

$$\sum_{i=1}^n (2i - 1) = 2 \sum_{i=1}^n i - \sum_{i=1}^n 1 = 2 \frac{n(n+1)}{2} - n = n^2 + n - n = n^2. \quad (2)$$

4 Linked List (3 points)

What is the output of this program?

```

#include <stdio.h>
#include <stdlib.h>
typedef struct listnode
{
    struct listnode * next;
    int value;
} Node;

Node * Node_construct(int v)
{
    Node * n = malloc(sizeof(Node));
    n -> value = v;
    n -> next = 0;
    return n;
}

void Node_destruct(Node * n)

```

Name:

7

Seat:

```

{
    free (n);
}

void List_destruct(Node * n)
{
    Node * prev = n;
    Node * next;
    while (prev != 0)
    {
        next = prev -> next;
        Node_destruct(prev);
        prev = next;
    }
}

void Node_print(Node * n)
{
    printf("%d ", n -> value);
}

void List_print(Node * n)
{
    Node * curr = n;
    while (curr != 0)
    {
        Node_print(curr);
        curr = curr -> next;
    }
    printf("\n\n");
}

Node * List_insert(Node * n, int v)
{
    Node * p;
    p = Node_construct(v);
    if (n == 0) /* original list is empty */
    {
        return p;
    }
    if ((v % 2) == 1) /* v is odd */
    {
        p -> next = n;
        return p;
    }
}

```

```

else /* v is even */
{
    Node * head;
    Node * prev;
    Node * curr;
    head = n;
    prev = n;
    curr = n;
    while (curr != 0)
    {
        prev = curr;
        curr = curr -> next;
    }
    /* prev cannot be 0 since n cannot be zero */
    p -> next = prev -> next;
    prev -> next = p;
    return head;
}
return 0; /* should never reach here */
}

int main(int argc, char * argv[])
{
    int cnt;
    Node * list = 0;
    for (cnt = 0; cnt < 10; cnt++)
    {
        list = List_insert(list, cnt);
    }
    List_print(list);
    List_destruct(list);
    return 0;
}

```

Answer:

9 7 5 3 1 0 2 4 6 8