Name: ____________________
U#: ____________________

Exam 3
COP 3514 Program Design

07/12/2013
2:00-3:15 pm

- **Closed books, notes, laptop, phone, neighbors**
- **malloc fail check required for all questions**
- **Good luck!**

1. Write a program that prompts the user to enter a number **N**. It then asks the user to enter **N** double numbers and stores them in an array. Memory for the numbers should be allocated dynamically. After the user enters **N** numbers the elements of the array should be printed in reverse order and the memory should be deallocated.

   ***NOTE: No input validation required.**

(25 points)

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

int main()
{
        double* array=NULL;
        int N,i;
        printf("Please enter N\n");

        scanf("%d",&N);

        array=(double*) malloc(N*sizeof(double));

        if(array==NULL)
                return -1;

        for(i=0;i<N;i++)
                scanf("%lf",&array[i]);

        for(i=(N-1);i>=0;i--)
                printf("%lf ",array[i]);
        printf("\n");

        free(array);
        array=NULL;
        return 0;
}
```

2. **a)** Allocate memory for **p1** and **p2**. Set the **first_name** of **p1** to “**Channing**” and the **last_name** to “**Tatum**”. Set the **first_name** of **p2** to “**Jenna**” and the **last_name** to “**Dewan**”. If a **malloc** fails return -1.

(5 points)

```
typedef struct
{
        char* first_name;
        char* last_name;
}Person

Person* p1=NULL;
Person* p2=NULL;


p1=(Person*) malloc (sizeof(Person));
if(p1==NULL)
        return -1;

p1->first_name=(char*) malloc (strlen(“Channing”)+1);
if(p1->first_name==NULL)
        return -1;
strcpy(p1->first_name,”Channing”);

p1->last_name=(char*) malloc (strlen(“Tatum”)+1);
if(p1->last_name==NULL)
        return -1;
strcpy(p1->last_name,”Tatum”);

p2=(Person*) malloc (sizeof(Person));
if(p2==NULL)
        return -1;

p2->first_name=(char*) malloc (strlen(“Jenna”)+1);
if(p2->first_name==NULL)
        return -1;
strcpy(p2->first_name,”Jenna”);

p2->last_name=(char*) malloc (strlen(“Dewan”)+1);
if(p2->last_name==NULL)
        return -1;
strcpy(p2->last_name,”Dewan”);
```

b) Complete the function below that deallocates the memory for the pointer **ChanningTatum**

(5 points)

```
void G_I_Joe_Retaliation(Person * ChanningTatum)
{

        free(ChanningTatum->first_name);
         ChanningTatum->first_name=NULL;

        free(ChanningTatum->last_name);
         ChanningTatum->last_name=NULL;

        free(ChanningTatum);
         ChanningTatum=NULL;    // :(

}
```

c) Complete the function bellow such that it changes the **last_name** of **p2** to the **last_name** of **p2** and **p1**. For example if the **last_name** of **p2** is "**Dewan**" and the **last_name** of **p1** is "**Tatum**" the new **last_name** of **p2** should be "**Dewan-Tatum**". The function also creates a new variable called **baby** which is a pointer to a **Person** structure and sets the **first_name** of **baby** to "**Everly"** and sets the **last_name** to the **last_name** of **p1**. The function returns the pointer **baby**.

(15 points)

```
Person* Marry(Person* p1, Person* p2)
{

        char* temp=realloc(p2->last_name,strlen(p2->last_name)+strlen(p1->last_name)+1);
        if(temp==NULL)
                printf("ERROR");

        p2->last_name=temp;
        strcat(p2->last_name,"-");
        strcat(p2->last_name,p1->last_name);

        Person* baby=NULL;


        baby=(Person*) malloc (sizeof(Person));
        if(baby==NULL)
                printf("ERROR");

        baby->first_name=(char*) malloc (strlen("Everly")+1);
        if(baby->first_name==NULL)
                printf("ERROR");
        strcpy(baby->first_name,"Everly");

        baby>last_name=(char*) malloc (strlen(p1->last_name)+1);
        if(baby->last_name==NULL)
                printf("ERROR");
        strcpy(baby->last_name,p1->last_name);

        return baby; // :)
}
```

3. Complete the **ShiftLeft()** function such that it shifts all of the nodes in a linked list to the left, the last node is set to the first node of the list.

   NOTE: The function must shift the entire node not the info parameter of a node.

   Example:
   A list before the function call
   4 3 7 9 2
   The same list after the function call
   3 7 9 2 4

(25 points)

```
void ShiftLeft(Node** list)
{


	Node* first = *list;

	Node* last =*list;

	while(last->next!=NULL)
	{
		last=last->next;
	}
	last->next=first;


	*list=first->next;
	first->next=NULL;

}
```

4. Write a program that reads numbers from a **Input.txt** file and writes them in reverse order in a **Output.txt** file. Memory for the numbers should be allocated dynamically. (Hint: use **realloc**).

   Example:

Input.txt	Output.txt
1	**4**
2	**3**
3	**2**
4	**1**

(25 points)

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

int main()
{
        FILE* f=fopen("Input.txt","r");
        FILE* f1=fopen("Output.txt","w");

        int* array=malloc(sizeof(int));
        int index=0;
        int number;
        while(fscanf(f,"%d\n",&number)==1)
        {
                int* temp=realloc(array,(index+1)*sizeof(int));
                if(temp==NULL)
                        return -1;

                array=temp;
                array[index]=number;
                index++;

        }

        int i;
        for(i=(index-1);i>=0;i--)
        {
                fprintf(f1,"%d\n",array[i]);
        }
        fclose(f);
        fclose(f1);
        return 0;
}
```

Bonus question:
You will receive partial credit on the bonus question only if it is more than 50% accurate.

(10 points)

You are given:

```
typedef struct Dream
{
        int number;
        struct Dream* dream;
}Dream;

Dream *first_dream;
```

Complete the ***recursive*** **createInception()** function such that it creates N dream structures linked together. The first dream structure has a number value N, the second one has a number value of N-1 and so on. Assume that the function is called as **createInception(&first_dream).** Each time a dream is created a "**Dream created**" should be written to Output.txt. For example if 3 dream structures are created the Output.txt should look like this:

```
Dream created
Dream created
Dream created
```

```
//Must use recursion
void createInception(Dream** d,int N)
{

        FILE *f=fopen("Output.txt","a");
        *d=malloc (sizeof(Dream));
        (*d)->number=N;
        fputs("Dream created\n",f);
        fclose(f);
        createInception(&((*d)->dream),N-1);
}
```