Sean Poston

CS351 Homework 8

11.11:

a)

```
15          struct person person = {"unassigned", "", "0"};
16
17          for (int i = 0; i < 100; i++) {
18              fwrite(&person, sizeof(struct person), 1, cfPtr);
19              fprintf(cfPtr, "\n");
20          }
```

b)

```
14      else {
15          char lastName[15];
16          char firstName[15];
17          char age[4];
18          for (int i = 0; i < 10; i++) {
19              puts("Enter lastName: ");
20              scanf("%s", &lastName);
21              puts("Enter firstName: ");
22              scanf("%s", &firstName);
23              puts("Enter age: ");
24              scanf("%s", &age);
25
26              struct person person = {lastName, firstName, age};
27
28              fwrite(&person, sizeof(struct person), 1, cfPtr);
29          }
```

c)

```
14      else {
15          int row;
16
17          struct person person = {"unassigned", "", "0"};
18
19          printf("%s", "Enter row to update");
20          scanf("%d", row);
21
22          fseek(cfPtr, (row - 1) * sizeof(struct person), SEEK_SET);
23
24          fread(&person, sizeof(struct person), 1, cfPtr);
25
26          if (person.lastName = "unassigned") {
27              puts("No information");
28          }
29          else {
30              puts("Enter new first name: ");
31              scanf("%s", &person.firstName);
32              puts("Enter new last name: ");
33              scanf("%s", &person.lastName);
34              puts("Enter new age: ");
35              scanf("%s", &person.age);
36
37              fseek(cfPtr, (row - 1) * sizeof(struct person), SEEK_SET);
38              fwrite(&person, sizeof(struct person), 1, cfPtr);
39          }
```

d)

```
14      else {
15          int row;
16
17          struct person person = {"unassigned", "", "0"};
18
19          printf("%s", "Enter row to update");
20          scanf("%d", row);
21
22          fseek(cfPtr, (row - 1) * sizeof(struct person), SEEK_SET);
23
24          fread(&person, sizeof(struct person), 1, cfPtr);
25
26          if (person.lastName = "unassigned") {
27              puts("No information");
28          }
29          else {
30              fseek(cfPtr, (row - 1) * sizeof(struct person), SEEK_SET);
31              fwrite(&person, sizeof(struct person), 1, cfPtr);
32          }
```

12.8: https://repl.it/@seanposton4/advancedRandom

```
The sum is: 1222
The average is: 48.880000

Current List:
0 1 4 4 11 20 23 27 28 29 29 34 38 55 65 68 71 76 78 88 92 93 94 96 98
RUN SUCCESSFUL (total time: 60ms)
```

Source Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

struct listNode {

   char data;

   struct listNode *nextPtr;

};


typedef struct listNode ListNode;

typedef ListNode *ListNodePtr;


void insertionSort(int * list) {

   int j = 1;

   int temp;

   for (int i = 1; i < 25; i++) {

      j = i;

      while (j > 0 && list[j - 1] < list[j]) {

         temp = list[j - 1];

         list[j - 1] = list[j];

         list[j] = temp;

         j--;

      }

   }
```

```
}

void insert(ListNodePtr *sPtr, int value);
void sumList(ListNodePtr currentPtr);
void printList(ListNodePtr currentPtr);

int main(void) {
   time_t t;
   srand(time(&t));
   int list[25];

   for(int i = 0; i < 25; i++)
      list[i] = (rand() % 100);
   insertionSort(list);

   ListNodePtr startPtr = NULL;

   for (int i = 0; i < 25; i++) {
      insert(&startPtr, list[i]);
   }
   sumList(startPtr);
   puts("");
   printList(startPtr);
   return 0;
}

void insert(ListNodePtr *sPtr, int value) {
   ListNodePtr newPtr;
   ListNodePtr previousPtr;
```

```
   ListNodePtr currentPtr;

   newPtr = malloc(sizeof(ListNode));

   if(newPtr != NULL) {
      newPtr->data = value;
      newPtr->nextPtr = NULL;

      previousPtr = NULL;
      currentPtr = *sPtr;

      while (currentPtr != NULL && value > currentPtr->data) {
         previousPtr = currentPtr;
         currentPtr = currentPtr->nextPtr;
      }

      if (previousPtr == NULL) {
         newPtr->nextPtr = *sPtr;
         *sPtr = newPtr;
      }
      else {
         previousPtr->nextPtr = newPtr;
         newPtr->nextPtr = currentPtr;
      }
   }
   else {
      puts("no.");
   }
}
```

```
void sumList(ListNodePtr currentPtr) {
   int sum = 0;

   while(currentPtr != NULL) {
      sum += currentPtr->data;
      currentPtr = currentPtr->nextPtr;
   }

   printf("The sum is: %d\nThe average is: %f\n", sum, (float)sum / 25.0);
}

void printList(ListNodePtr currentPtr) {
   puts("Current List:");
   while (currentPtr != NULL) {
      printf("%d ", currentPtr->data);
      currentPtr = currentPtr->nextPtr;
   }
}
```