Practice Test 3, Program Design

1. Assume that the node structure is declared as:

```
struct node {
    int value;
    struct node *next;
};
```

The following function returns the number of nodes that contains `n`; it returns 0 if n doesn't appear in the list. The `list` parameter points to a linked list.

```
int count_n(struct node *list, int n){
    struct node *p;
    int count = 0;
    for(p = list; p != NULL; p=p->next) {

        //missing statement(s)

    }
    return count;
}
```

Write statement(s) in place of *missing statement(s)* that will compare the value of the nodes in the linked list with `n` and update count if the value equals to `n`.

Answer:

2. Suppose you have just opened the file test.txt and execute the statements shown below:

```
char buffer[6];
...
fscanf(pFile, "%s", buffer);
printf ("%s", buffer);
```

File test.txt consists of a single line of text:

```
The quick brown fox jumped over the lazy dog's back.
```

What will be the output from the printf?

Answer:

3. Below is a program including a recursive function.

```
#include <stdio.h>
void countdown(int n) {
      printf ("n = %d\t", n);
      n--;
      if (n >= 0) {
            countdown(n);
      }
      printf ("n = %d\t", n);
}

int main ( ) {
      countdown(2);
      return 0;
}
```

What will be the output when this program is run?

Answer:

4. A program intended to find the maximum value in an array without changing the array is shown below. The final output is correct but it unintentionally modifies the array. Which line causes the modification?

```
1) #include <stdio.h>
2) int main()
3) {
4)     int* pMax;
5)     int* pNext;
6)     int numbers[] = {33, 15, 65, 98, 21, 0, -12};
7)     pMax = numbers;
8)     pNext = numbers;
9)
10)          while (pNext <= &numbers[6])
11)          {
12)              if (*pNext > *pMax)
13)              {
14)                  *pMax = *pNext;
15)              }
16)              pNext++;
17)          }
18)          printf ("Max = %d\n", *pMax);
19)          return 0;
20)      }
```

Answer:

5. Write a statement that declares a dynamic allocated array a of integers of size `n` using malloc function.

Answer:

6. What is the output of the following program?

```
#include <stdio.h>
int f1(int (*f) (int));
int f2(int i);
int main(void) {
    printf("%d\n", f1(f2));
    return 0;
}
int f1(int (*f) (int)) {
    int n, num = 0;
    for(n = 0; n < 3; n ++)
        num = f(n);
    return num;
}
int f2(int i) {
    return i * i + i;
}
```

Answer:

7. Suppose you have just opened a student data file with the following data formats

```
84652123 John 12 11 1995
17680087 Mary 9 21 1997
…(more data)
```

The first field in a row is the student id, the second is the student first name, the third is the month of the student's birthday, and fourth is the day of the student's birthday and the fifth is the year of the student's birthday. What is the output of the following statements? Assume `id` is an int variable, `first_name` is a string variable, `month`, `day`, and `year` are int variables, `value` is an int variable.

```
value = fscanf(pFile, "%d%s%d%d%d", &id, first_name, &month, &day, &year);
printf("%d", value);
```

Answer:

8. Given a linked list of node declared as follows, how would you know if a linked list of `node` named `list` is an empty list?

```
struct node {
   int number;
   struct node *next;
};
```

Answer:

9. A function named `integrate` that integrates a mathematical function `f` can be made by passing `f` as an argument, and `a` and `b` are the range for the integration, both and `a` and `b` are of type double. Function `f` can be any function that takes a double parameter and returns a double. The function `integrate` returns a double. Write a function prototype for the `integrate` function.

Answer:

10. What is the problem with the following code? Assume list represents a linked list and the function is supposed to release the memory occupied by the nodes in the linked list.

```
void clear_list(struct node *list) {
    free(list);
}
```

a) Memory leak
b) Dangling pointer
c) The function is correct.

11. Which of the following loops correctly uses p to move through the nodes of the linked list pointed by `list`? Assume that the structure is declared as:

```
struct node {
      int value;     /* data stored in the node  */
      struct node *next;
                      /* pointer to the next node */
    };
    a) while (p != NULL) p++;
    b) while (p != NULL) p = p-> next;
    c) for (p = NULL; p != NULL; p = p->next)
    d) for (p = list; p != NULL; p = p->next)
```

Answer:

12. The following function is supposed to delete the first node containing value `n` in a linked list, returning a pointer to the first node in the modified list. Which of the following functions will delete the node correctly? Assume that the structure `node` is defined as:

```
struct node {
   int value;
   struct node *next;
}

  a)
struct node *delete(struct node *list, int n) {
   struct node *cur, *prev;
   for(cur = list, prev = NULL;
         cur!= NULL && cur->value!= n;
         prev = cur, cur = cur->next)
         ;

   if (cur == NULL)
         return list;
   if (prev == NULL)
         list = list->next;
   else
         prev->next = cur ->next;
   free(cur);
   return list;
}

  b)
struct node *delete(struct node *list, int n) {
   struct node *cur, *prev;
   for(cur = list, prev = NULL;
         cur!= NULL && cur->value!= n;
         prev = cur, cur = cur->next)
```

```
            ;

    if (prev == NULL)
          return list;
    if (cur == NULL)
          list = list->next;
    else
          prev->next = cur ->next;
    free(cur);
    return list;
 }
```

c) None of the above.

Answer:

13. The following function is supposed to insert a new node into its proper place in an ordered list, returning a pointer to the first node in the modified list. Which of the following functions will work correctly in all cases? Assume that the node structure is defined as:

```
    struct node {
          int value;
          struct node *next;
    }
```

a.

```
struct node *insert_into_ordered_list(struct node *list, struct node
*new_node)
{
    struct node *cur = list, *prev = NULL;
    while(new_node->value > cur->value){
          prev = cur;
          cur = cur->next;
    }

    new_node->next = cur;
    prev->next = new_node;
    return list;
 }
```

b.

```
 struct node *insert_into_ordered_list(struct node *list, struct node
 *new_node)
 {
    struct node *cur, *prev;
```

```
   for(cur = list, prev = NULL;
         cur!= NULL && new_node->value > cur->value;
         prev = cur, cur = cur->next)
         ;


         if(prev != NULL){
               prev->next = new_node;
               new_node->next = cur;
               return list;
         }
         else
               return new_node;

}
```

c.

```
struct node *insert_into_ordered_list(struct node *list, struct node
*new_node)
{
   struct node *cur, *prev;
   for(cur = list, prev = NULL;
         cur!= NULL && new_node->value > cur->value;
         prev = cur, cur = cur->next)
         ;

   new_node->next = cur;
   prev->next = new_node;
   return list;

}
```

d.

```
struct node *insert_into_ordered_list(struct node *list, struct node
*new_node)
{
   struct node *cur, *prev;
   for(cur = list, prev = NULL;
         cur!= NULL && new_node->value > cur->value;
         prev = cur, cur = cur->next)
         ;

         new_node->next = cur;
         if(prev == NULL)
               return new_node;
         else{
               prev->next = new_node;
```

```
            return list;
        }
}
```

Answer:

14. The call to qsort must pass in a function pointer for a compare function to be used in sorting the array. The function header for qsort is

```
void qsort (void* base,              //array to be sorted
            size_t n,                //length of array
            size_t size,             //size of each entry
            int (*cmp) (const void *p, const void *q))
                                    //function pointer for compare function
```

Which of the following function can be passed to qsort in order to sort an integer array?

```
a) int int_cmp(const void *p, const void *q) {
      return *p - *q;
     }
b)  int int_cmp(const void *p, const void *q) {
      return *(int *)p - *(int *)q;
     }
c)  int int_cmp(const void *p, const void *q) {
      return (int *)p - (int *)q;
     }
d)  int int_cmp(const void *p, const void *q) {
      return p - q;
     }
```

Answer:

15. Suppose you have just opened a student data file with the following data format:

```
John 12 11 1995
Mary 9 21 1997
… (more data)
```

The first field is the student's name (one word string), and the next three are respectively the month, the day, and the year of the student's birthday (integers). Complete the fscanf statement below to read one line of the file and store the obtained values in the variables below: Assume the FILE pointer is fp for the opened file.

```
int month, day, year;
char name[101];
```

The fscanf statement must work when used in a loop to read consecutive lines of the file.

Answer:

16. Which of the alternatives completes the function clear_list to release all nodes from a linked list passed as argument?

```
struct node {
  int value;
  struct node *next;
};
void clear_list(struct node *list) {
  struct node *temp;
  while(list != NULL) {
    temp = list;
    // missing code here
  }
}
```

```
A) list=temp->next;
   free(temp);

B) free(temp);
   list=list->next;

C) free(list);
   list=temp->next;

D) list=list->next;
   free(list);
```

Answer:

17. Given the opt structure below, which of the following statements correctly creates one opt node using dynamic memory allocation?

```
struct opt {
  char option;
  struct opt *next;
};
struct opt *new_node;
```

```
A)
new_node = malloc(sizeof(struct opt));

B)
new_node = malloc(sizeof(char) + sizeof(struct opt *));

C)
new_node = malloc(sizeof(char)) + malloc(sizeof(struct opt *));

D)
struct opt node; new_node = &node;
```

Answer:

Free-From Questions

18. Complete the following function:

```
int count_characters(const char *filename);
```

The function should open, read, and close the file, and return the number of characters in the text file whose name is `filename`. If there is no character in the file or the file does not exit, the function should return 0. Assume the maximum number of characters in each line is 1000.

You might need the file reading function `fgets`. The function prototype of `fgets` is:

```
char* fgets (char* buffer, int max, FILE* pFile)
```

```
int count_characters(char *filename)
{
  char str[1001];
  FILE* pFile;
  int count = 0;
}
```

19. Complete the `search` function that looks up the part in *`inv`* (an array of struct part). The function prompts the user to enter a part number. If the part exists, prints the name and quantity on hand; if not, prints a “part not found” message. Parameter *`np`* contains the number of parts in the array.

    Assume the structure `part` is defined as:

```
struct part{
  int number;
  char name[31];
  int on_hand;
};

void search(struct part inv[], int np)
{     // add additional variable declarations if necessary



      int number;
      printf(“Enter part number: “);
      scanf(“%d”, &number);
}
```

20. Write a function that finds the derivative of a function `f` at a value `x`. It takes a function pointer `f` as argument that represents the function for computing the derivative. The second parameter for the function is a double that represents value x. The function returns a double.

    The derivative of a function can be calculated by

    $$f'(x) = \frac{f(x+stepSize)-f(x-stepSize)}{2*stepSize}$$

    Use 0.01 for `stepSize`.

21. Write the following function:

```
struct node *move_last_to_first(struct node *list);
```

The list parameter points to a linked list of the following structure.

```
struct node {
      int value;
      struct node *next;
}
```

The function should move the last node to be the head of the list, before the first node. The function should return a pointer to the head of the updated linked list.

22. After implementing a function to delete the first node containing the value n from a linked list, you noticed that you could call this function several times to delete all nodes containing the value n. Given this delete function and a function size that returns the number of nodes in a linked list, complete the function delete_all below. Hint: delete nodes with value n from the list until the number of nodes in the list does not change.

```
struct node {
  int value;
  struct node *next;
};
struct node *delete(struct node *list, int n) {
  struct node *cur, *prev;
  for(cur=list, prev=NULL; cur!=NULL && cur->value!=n; prev=cur,
cur=cur->next) ;
  if(cur == NULL)
    return list;
  if(prev == NULL)
    list = list->next;
  else
    prev->next = cur->next;
  free(cur);
  return list;
}
int size(struct node *list) {
  int r=0;
  for(struct node *p=list; p!=NULL; p=p->next, r++) ;
  return r;
}

struct node *delete_all(struct node *list, int n) {









}
```

23. Write a program that APPENDS the sentence "That's all, folks!" to a file provided as a command-line argument (Hint: open the file in append mode). For instance, if the executable file created from your code is named a.out, the command below must append the sentence above to the file "looney_tunes.txt":

```
./a.out looney_tunes.txt
```

```
#include <stdio.h>
int main(int argc, char *argv[]) {
  FILE *fp;










  return 0;
}
```