University of Toronto
Faculty of Applied Science and Engineering

Final Examination, April 2013

First Year - APS 105

APS105H1

Exam Type: A
"This is a closed book examination, no aids are permitted"

Examiner - Abbas. A

Question	**Mark**
1) Short Answer	/20
2) Sorting and Search	/15
3) Loop Invariant	/15
4) Recursion	/15
5) Strings	/15
6) Linked List	/20

THIS EXAM CONTAINS TOTAL OF 18 PAGES INCLUDING THE APPENDIX AND THE FRONT COVER.

Question 1) Short Answers **[20]**

a) What are *recursive* functions in C? [1]

b) "Every recursive function has atleast *one base/terminating case*". Why is this important? [2]

c) What is *memory leak*? Give one example of memory leak [2]

d) How is memory ***deallocated*** on the heap? How is memory ***deallocated*** on the stack? Give one example of each. [2]

e) You like to search for the name *"Matthews"* in a telephone book directory (with names starting from A to Z). Will *linear search* be faster than *binary search*? Why or why not? In general, what is the *worst-case performance* (i.e. # of loop iterations) for *linear search* and *binary search*? [3]

f) What will the following code print, on a 64 **AND** 32 bit machine? [5]

```
#include <stdio.h>
int main()
{


  printf("sizeof on int is %lu bytes\n",sizeof(int));
  printf("sizeof on char is %lu bytes\n",sizeof(char));
  printf("sizeof on double is %lu bytes\n",sizeof(double));
  printf("sizeof on float is %lu bytes\n",sizeof(float));

  printf("sizeof on int* is %lu bytes\n",sizeof(int*));
  printf("sizeof on char* is %lu bytes\n",sizeof(char*));
  printf("sizeof on double* is %lu bytes\n",sizeof(double*));
  printf("sizeof on float* is %lu bytes\n",sizeof(float*));
  return 0;

}
```

g) Your instructor has written following code to print out the string "abc" as shown below. Unfortunately it is incorrect.

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

int main()
{
    char *p;
    p[0]='a';
    p[1]='b';
    p[2]='c';
    p[3]='\0';
    printf("the string content is %s",*p);

}
```

g.1) Why is the above code incorrect? [2]

g.2) Modify and correct the above code, such that the string *"abc"* is printed. [3]

Question 2) Sorting (Ascending Order) and Search **[15]**

YOU ARE NOT REQUIRED TO WRITE ANY CODE FOR THIS QUESTION

Given the following array of integers for a), b), c), d) and e):

2	*100*	*-1*	*50*	*3*

a) Perform <u>Bubble Sort</u> (clearly show all the steps in order to receive full credit). [3]

b) Perform <u>Selection Sort</u> (clearly show all the steps in order to receive full credit) [3]

c) Perform Quick Sort (clearly also mark your ***pivot element*** and the output of the ***partition function*** in order to receive full credit). [4]

d) Perform Linear Search i.e. search for the integer -999 (clearly show all the steps in order to receive full credit) [2]

e) Perform Binary Search i.e. search for the integer -999 (clearly show all the steps (including *left marker, right marker* and *middle marker*) in order to receive full credit) [3]

Question 3) Loop Invariant. **[15]**

a) Define *'loop invariant'* [2]

b) Write a function called *multiplicationOfArrayValues* that will return back the multiplication of every single value in the array of integers. *i.e. assuming that the array is of length N then mul=A[0]*A[1]*...*A[N-1]*

b.1) What is the *loop invariant* for this problem? [5]

b.2) Complete the following function using *while loop* that meets the specification of b) above. **<u>You MUST also mark/highlight where in the code your 'loop invariant' is TRUE.</u>** [8]

```
int multiplicationOfArrayValues (                                )
{



}
```

Question 4) Recursion. **[15]**

a) The function declaration for the *power*(...) function is defined as: [6]

```
int power(int x, int n);
```

Example:

`power(2, 3)` will return 8 i.e. 2^3
`power(3, 2)` will return 9 i.e. 3^2
`power(4, 0)` will return 1 i.e. 4^0

Note: If n is a power of 2, then x^n can be computed by squaring. For example, x^4 is the square of x^2, so x^4 can be computed i.e. $(x^2)*(x^2)$. This technique can be used also when n is not a power of 2. If n is even, use the formula $x^n=(x^{(n/2)})^2$. If n is odd, then use $x^n=x*x^{n-1}$. The range for n is >=0.

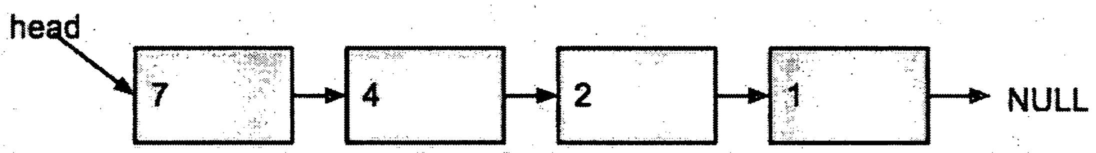
Write a recursive function that computes x^n.

```
int power(int x, int n)
{









}
```

b) Write a *recursive* function called *freeSimpleLinkedList(...)* that takes in as an input argument a head pointer of type *struct node* and frees/deallocates every node in that simple linked list from the heap. [5]

```
void freeSimpleLinkedList(struct node* head)
{











}
```

c) Trace out the recursive function call of your solution b) above by drawing the memory diagram using Stack and Heap for the following linked list. [4]



Question 5) Strings **[15]**

a) Write a function that test whether two words are anagrams (permutations of the same letters). For example "*smartest*" and "*mattress*" are anagrams. On the other hand "*dumbest*" and "*stumble*" are not anagrams. If the two words are anagrams, your function must return back TRUE otherwise it will return back FALSE. [8]

```
bool are_anagrams (const char *word1, const char *word2)
{
















}
```

b) [7]

Write a function named *duplicate* that uses *dynamic storage allocation* to create a copy of a string. For example, the call

```
char *str="abc";
char *p=duplicate(str);
```

would allocate space for a string of the same length as *str*, copy the contents of *str* into the new string, and return a pointer to it. Have *duplicate* return a null pointer if the memory allocation fails.

```
char* duplicate(const char* s)
{











}
```

Question 6) Linked List. **[20]**

a) Write a function that checks whether a *doubly linked list (every node contains two pointers i.e. next pointer and a previous pointer)* is a palindrome (the characters in the doubly linked list are the same from left to right as from right to left). *You can safely assume that the doubly linked list contains characters that are all lower case and contains no numbers or white space. The doubly linked list can be of any length>=0. The HEAD pointer points to the first node of your doubly linked list, and the TAIL pointer points to the last node of your doubly linked list.* [5]

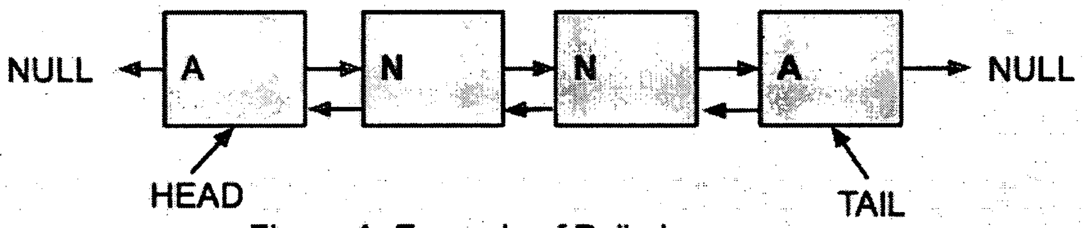


Figure A: Example of Palindrome

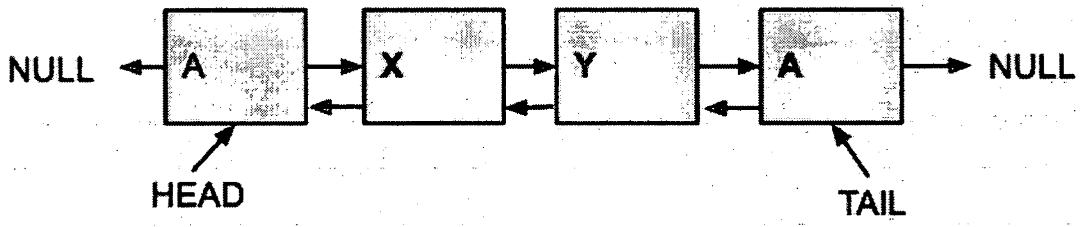
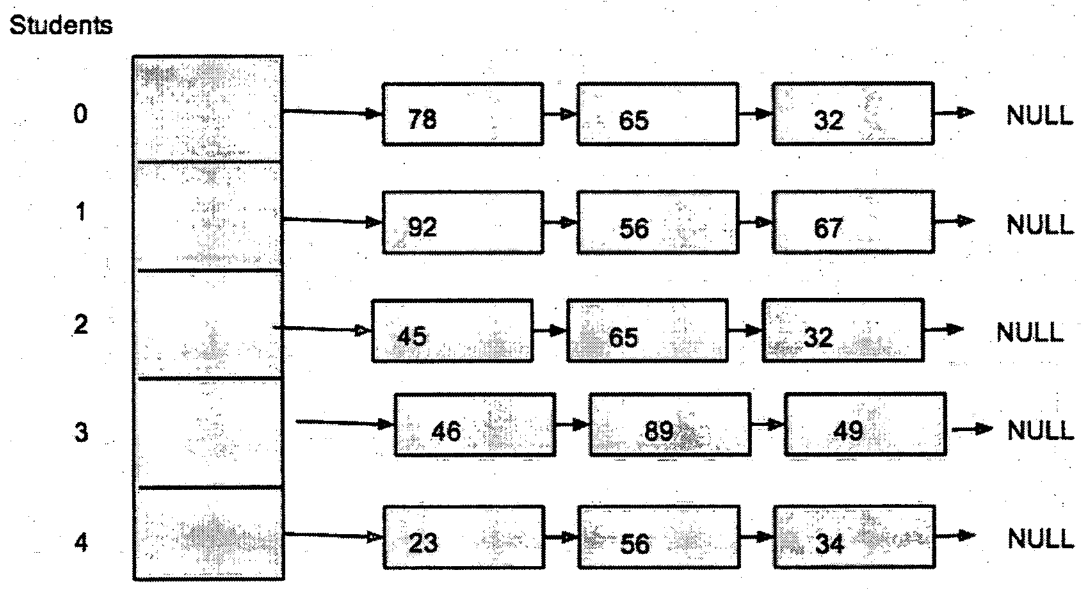


Figure B: Example of not a Palindrome

YOUR SOLUTION COMES ON THE NEXT PAGE

```
void isPalindrome(struct dnode* HEAD, struct dnode* TAIL)
{













}
```

b) A certain class at University of Toronto has only 5 students. This class for the entire semester has 3 quizzes in total where each quiz is out of 100 marks. The professor of this class, decided to maintain an *array of linked list* to store the quiz marks for each student. ***One such example*** is show in the figure below where *Students* is an array of size 5. Each student is labeled as (0,1,2,3,4) and has a linked list of size 3 that stores the quiz marks for that student. For example, the (quiz3, quiz2, quiz1) marks for student0 are (78,65,32). Each quiz node is of type *struct quiz_mark. You can assume that Students is a GLOBAL variable.*



b.1) What is the definition of Students? CHOOSE ONE CORRECT ANSWER FROM THE FOLLOWING 5 OPTIONS [2]

- Option A) int Students [5];
- Option B) int *Students [5];
- Option C) struct quiz_mark *Students [5];
- Option D) struct quiz_mark Students [5];
- Option E) Array_Of_Linked_List(sizeof(struct quiz_mark)*5);

b.2) Write the following function that creates a new *struct quiz_mark* node on the heap and is also inserted at the front of the correct linked list of that student (i.e. the new quiz mark becomes the first node of that linked list). PRINT AN APPROPIATE ERROR MESSAGE IF THE SIZE OF THE LINKED LIST EXCEEDS 3. *You can assume that Students is a GLOBAL variable. The studentNumber can be any of the following (0,1,2,3,4). The quizMark can be any number between 0 and 100.* [5]

```
void insertMark(int quizMark,int studentNumber)
{









}
```

b.3) Write the following function that prints the average mark for a specific student. Assuming the example in the figure (on the previous page) and *studentNumber*=0, your function will print the following. *You can assume that Students is a GLOBAL variable.* [5]

Average mark for student0 is 58.33

```
void printAverageForSINGLEStudent(int studentNumber)
{










}
```

b.4) Write the following function that prints the average mark for every student. This function MUST call the function that you implemented in b.3) above. [3]

The output of this function must be as follows:
Average mark for student0 is 58.33
Average mark for student1 is 71.66
Average mark for student2 is 47.33
Average mark for student3 is 61.33
Average mark for student4 is 37.66

```
void printAverageForAllStudent()
{










}
```

Appendix:

1)
```
struct node
{
    int value;
    struct node* next;
}
```

==

2)
```
struct dnode
{
    char value;
    struct dnode* next;
    struct dnode* previous;
}
```

==

3)
```
struct quiz_mark
{
    int value;
    struct node* next;
}
```

---

4)
function

strlen

```
size_t strlen ( const char * str );
```

Get string length

Returns the length of the C string *str*.

---

5)
function

strncat

```
char * strncat ( char * destination, char * source, size_t num );
```

Append characters from string

Appends the first *num* characters of *source* to *destination*, plus a terminating null-character.