COMP20003 Algorithms and Data Structures

Second Semester 2015
Final Examination
16/11/2015

Student Number: _____

Identical Examination papers: None.
Exam Duration: Three hours.
Reading Time: 15 minutes.
Length: This paper has 9 pages including this cover page.
Authorised Materials: English language dictionaries and foreign language dictionaries are the only authorised materials. Other paper and printed materials are not permitted. University regulations prohibit the use of electronic dictionaries. Calculators and other electronic devices: Calculators are not permitted.
Instructions to Invigilators: Students should be supplied with the examination paper and a script book. They may have unlimited script books on request. Students should write their ID number on the examination paper and on all script books. They may not remove any part of the examination paper from the room and must return any used and unused script books with their examination paper.
Library: The paper is not to be lodged with the Baillieu Library.
Instructions to Students: There are 6 questions. You should attempt all questions. Answers to all questions are to be written in your script book. You may use the reverse side of the page to make notes or prepare draft answers. The reverse sides will not be looked at or marked unless you clearly indicate that you wish this to be your final answer.
You may use pencil. This paper counts for 60% of your final grade. Marks shown are out of 60.

Grade Table (for teacher use only)

Question	Points	Score
1	13	
2	5	
3	6	
4	10	
5	16	
6	10	
Total:	60	

Programming

1. (13 points) This question is about string processing in the C programming language.

   A `palindrome` is a word or phrase that reads the same in both directions. For example, some palindromic words are: `civic`, `testset`, and `rotator`. Numbers, punctuation, and/or capital letters must also be symmetric in palindromic words, `e.g Ab2c2bA`. In palindromic phrases, however, spaces, capitalization, numbers, and punctuation may be asymmetric, and must therefore be removed before the phrase reads exactly the same in both directions, for example: `A man, a plan, a canal, Panama!; a Toyota; and Drab as a fool, aloof as a bard!`

   Your task in this question is to construct a `C` function that takes as input a string (possibly containing white space and/or punctuation), and outputs `TRUE` if the string is a palindrome and `FALSE` if the string is not a palindrome. For full marks, your function should be able to detect palindromic phrases, that is it should ignore whitespace, numbers, punctuation, and capitalization.

   You are given scaffolding code that defines `TRUE` and `FALSE`, reads in strings from stdin, contains a prototype for the function `checkpalindrome()` that you are to write, and makes a call to this function.

   You may call on any functions from the C standard library that you think would be useful. The scaffolding includes header files for the ctype and string libraries. Possibly useful functions declared in `ctype.h` include `isalpha(c)`, `isalnum(c)`, `isupper(c)`, and `isspace(c)`, which check whether `c` is an alphabetic character, an alphabetic or numeric character, upper case, or whitespace, respectively. The parameter `c` is an `int`, whose value can be represented as an `unsigned char`. The return value for each of these functions is `int` (non-zero if c satisfies the condition, zero if not). For example:

   ```
   if (isalpha('a')) printf("'a' is alphabetic\n");
   ```

   Other useful functions declared in `ctype.h` include `toupper(c)` and `tolower(c)`, which convert lower case alphabetic characters to uppercase and vice versa.

   You are reminded that the library function `char *fgets(char *s, int n, FILE *stream)`, used in the scaffolding `main()` function, reads at most $n-1$ characters into array s, stopping as soon as a newline is encountered. Note that the newline is included in the array s, which is then terminated by "\0. The function `fgets()` returns `s`, or `NULL` if end of file or error occurs.

   Write the function `checkpalindrome()`, to fit in with the scaffolding code. You should write the function in your script book. Include brief comments in the code to explain

your reasoning and help with readability. Marking of your code will be based on accuracy, completeness, best practice use of C, and readability. For partial marks, write a function that detects palindromic words, but does not detect palindromic phrases.

```
/**
* Scaffolding for palindrome checker.
* COMP20003 Semester 2 2015
* Nov 2015
*/

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

#define MAXSTRING 256
#define TRUE 1
#define FALSE 0
#define MALLOC_ERROR 2

/* prototype for the function you will be writing */
int checkpalindrome(char *buffer);

/**
* main(). No inputs. Reads in from stdin, line-by-line, and calls
* function checkpalindrome. Prints the line if checkpalindrome
* returns TRUE.
*/
int main(int argc, char **argv)
{
  char *buffer;

  if((buffer = (char *)malloc(MAXSTRING))==NULL)
  {
    printf("malloc failed\n");
    exit(MALLOC_ERROR);
  }

  while((buffer=fgets(buffer,MAXSTRING,stdin))!=NULL)
    if(checkpalindrome(buffer)) printf("%s\n", buffer);

  return(TRUE);
}

/**
* FILL-IN: write documentation for this function in your script book
*/
int checkpalindrome(char *buffer)
{
  /* write the code for this function in your script book */
}
```

2. (5 points) In this question you are asked to read a small program, written in C, that contains an error. You are asked to identify the error and write C code that fixes the error.

   The programmer's intention was to read in strings from `stdin`, store them in an array. Once all the strings have been read and stored, the program was to output the number of strings read and the, one by one, in the order in which they were read in, to print the strings and their length. However, the program does not work as intended.

   Answer the following questions in your script book:

   (a) (1 point) Show the output of the code as shown, with the input:

   ```
   giraffes
   eat
   leaves
   ```

   (b) (1 point) Describe the programming error and briefly explain why it is a problem.

   (c) (3 points) Write C code that will fix the problem. Refer to line numbers where you would insert, remove or change code.

   For partial marks, outline in comments or pseudocode, rather than C code, how you would fix the problem.

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4
5  /* This code contains an error. Your task is
6  *  to find, describe, and fix the error.
7  * COMP20003 November 2015 */
8
9  #define OK 0
10 #define MAXSTRING 200
11 #define NUMBER 10
12 #define MALLOC_ERROR 2
13
14 int main (argc, argv)
15 {
16   char **B;
17   char buffer[MAXSTRING];
18   int i, strings=0, arraysize=NUMBER;
19
20   if((B = (char **)malloc(NUMBER * sizeof(char *))) == NULL)
21   {
22     printf("initial malloc error\n"); fflush(stdout);
23     exit(MALLOC_ERROR);
24   }
25
26   while((fgets(buffer,MAXSTRING,stdin))!=NULL)
27   {
```

```
28      if(strings+1>arraysize)
29      {
30        arraysize = 2*arraysize;
31        B = realloc(B, (arraysize)*sizeof(char *));
32      }
33
34      /* get rid of the final \n in buffer*/
35      buffer[strlen(buffer)-1] = '\0';
36      B[strings] = buffer;
37      strings++;
38    }
39
40    printf("Read %d strings:\n",strings);
41
42    for(i=0;i<strings;i++)
43      printf("\t%s\t %d\n", B[i], (int)strlen(B[i]));
44
45    return(OK);
46  }
```

Computational Complexity

3. (6 points) This question is about computational complexity

   For the remaining parts of this question you are required only to give an answer. You are not required to explain or justify your answers in this section. Each correct answer gives 1 point.

   1. What is the time complexity of inserting one item into an unsorted linked list of n items? Give the closest upper bound in big-O notation.
   2. What is the time complexity of sorting an array of n items using top-down merge-sort? Give the closest upper bound in big-O notation.
   3. What is the time complexity of sorting an array of n items using bottom-up merge-sort? Give the closest upper bound in big-O notation.
   4. What is the space complexity of sorting an array of n items using top-down merge-sort? Give the closest upper bound in big-O notation.
   5. What is the space complexity of sorting an array of n items using bottom-up merge-sort? Give the closest upper bound in big-O notation.
   6. Given a hash table of size m, containing n items, give the worst case complexity of searching for one item when collision resolution is achieved with chaining using balanced search trees (instead of linked lists). Give the closest upper bound in big-O notation.

Priority Queues

4. (10 points) This question is about priority queues.

One data structure used for priority queues is the `heap`. The heap is a complete tree, represented as an array, that follows the "heap condition". In a binary heap, each non-leaf node has two child nodes, except possibly the rightmost non-leaf node.

(a) (4 points) You are given the following array:

100	**12**	**79**	**11**	**13**	**77**	**11**	**87**	**25**

i. (3 points) Using the `bottom-up` heap construction method, put these items into the appropriate order for a binary min-heap (minimum key is at the root). Show your workings in either the array or the tree representation of the heap. Show your final answer for the heap in `both` the array `and` the tree representations.
For reduced marks, you may construct the heap by inserting one item at a time.

ii. (1 point) Why is bottom-up heap construction preferred method preferred over inserting the items one at a time? Be specific in your answer.

(b) (4 points) Dijkstra's algorithm for single source shortest paths on a directed, weighted graph uses a priority queue to get the next edge with the minimum weight. For each of the following implementations of a priority queue, give the least upper bound big-O complexity of Dijkstra's algorithm, in terms of V , the number of vertices in the graph, and E, the number of edges in the graph. Explain your answers `briefly`.

i. (2 points) A heap is used for the priority queue.

ii. (2 points) A linked list sorted by weight is used for the priority queue.

The main operations of Dijsktra algorithm are:

```
1  pq = makePQ(G);
2  while(!emptyPQ(pq))
3  {
4    u = deletemin(pq);
5    for(/*each vertex v reached from u */)
6      if(dist[u] + edgeweight(u,v) < dist[v])
7        update(v,pred,dist,pq);
8   }
```

(c) (2 points) A quaternary heap is a heap in which each non-leaf node in the tree representation has `four` children, except possibly the rightmost non-leaf node. In an array representation of a quaternary heap, what are the array indices of the children of the node at position i in the array? Be explicit about any assumptions or conditions you have made.

Graphs

5. (16 points) This question is about graphs and graph algorithms.
   (a) (3 points) Express the running time $\Theta()$ of the Floyd-Warshall algorithm for the all pairs shortest path problem for a graph $G(V, E)$:
      i. (1 point) In terms of the number of vertices V in G.
      ii. (1 point) In terms of the number of edges E in a dense graph G.
      iii. (1 point) In terms of the number of edges E in a sparse graph G.
   (b) (3 points) Which of the following are true and which are false? *Explain your answers*
      i. (1 point) Depth-first search expands at least as many nodes as A^* with $h(n) = 0$.
      ii. (1 point) $h(n) = 0$ for all nodes n is an admissible heuristic for the *8-puzzle*.
      iii. (1 point) Uniform Cost, also known as Dijkstra, is a special case of WA^* when $f(n) = g(n) + wh(n)$ and $w = 0$.
   (c) (5 points) Consider a connected weighted directed graph $G = (V, E, w)$. Define the `fatness` of a path P to be the `maximum weight` of any edge in P. Give an efficient algorithm that, given such a graph and two vertices $u, v \in V$, finds the `minimum` possible `fatness` of a path from u to v in G.
   (d) (5 points) Prim's algorithm constructs a Minimum Spanning Tree (MST) of a `connected`, weighted graph G(V, E). In this question you are asked to sketch a small enhancement to the traditional Prim's algorithm that would enable it construct a MST for each connected component of an `unconnected`, weighted graph G before terminating.

      For your convenience, a brief description of Prim's algorithm is given below, as described in lectures and in your textbook.

      In Prim's algorithm, the MST grows by adding a new vertex v from the "fringe", that is vertex v is not yet in the tree and also has an edge (u, v) to a vertex u that is already in the tree. Vertices in the fringe are stored in a priority queue, according to their least weight edge $w(u, v)$ to a vertex in the MST. The vertex with the minimum edge weight $w(u, v)$ is picked and removed from the priority queue. Once a vertex v is put into the MST, each vertex reachable from vertex v and not yet in the MST is added to the priority queue, or if already in the priority queue has its weight (priority) and predecessor updated if appropriate.

      Prim's algorithm can start with any vertex in G, and terminates when the priority queue is empty (or, depending on implementation when all vertices in the priority queue have weight ∞). Auxiliary arrays are used to keep track of predecessors, edge weights, and whether or not a vertex is in the MST.

Only a connected graph can have a minimum spanning tree, and Prim's algorithm works only on connected graphs. Vertices that cannot be reached from the growing MST are not inserted into the priority queue (or, depending on implementation, remain in the priority queue weight ∞ and are never updated or deleted).

Sketch briefly, in words or pseudocode, a small enhancement you could make to Prim's algorithm that would enable it to construct an MST for `each` connected component of an unconnected, weighted graph G before terminating.

Searching

6. (10 points) This question is about algorithms, data structures, and operations on nodes used in searching.

   (a) (5 points) You are given records with the following keys, to be stored in a hash table

   17 53 41 29

   The hash table contains a maximum of 12 items, and the hash function `hash(key)` returns $key\%12$. Linear probing is used for collision resolution.

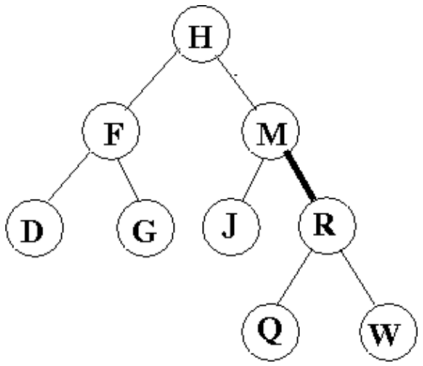
   i. (1 point) Show the hash table after insertion of these records

   ii. (1 point) How many key comparisons will be needed to search for the record with $key = 125$ in the table you have populated in part i above?

   iii. (3 points) State another method for collision resolution that would reduce the number of key comparisons needed to search for the record with $key = 125$ in this table and explain `briefly` how this method would achieve a lower number of comparisons.

   (b) (5 points) The rotation operation is used in AVL trees, red-black trees, and splay trees to adjust the balance of a binary tree.

   In the binary search tree shown below, you are to perform a left rotation of the edge between nodes `M` and `R`. This edge is shown as a thicker line.



   i. (2 points) Draw a diagram showing the configuration of all nodes in this tree after a left rotation between nodes `M` and `R`.

ii. (3 points) You are to write a series of statements in the C programming language that accomplish a left rotation. Use pointers to the `grandparent`, `parent`, and `child` nodes (`H`, `M`, and `R`, respectively in the diagram above), and the following definition of a node:

```
1      struct node *{
2        char key;
3        struct node *left;
4        struct node *right;
5      };
```