NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 2 EXAMINATION 2015-2016

CE1007/CZ1007 – DATA STRUCTURES

April/May 2016 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 9 pages.
2. Answer **ALL** questions.
3. This is a closed-book examination.
4. All questions carry equal marks.

---

1. (a) What is the output of the following code?

```
#include <stdio.h>
void f1(int y, int *x);
int main() {
   int x=2,y=4,i;
   for (i=0; i<2; i++) {
      f1(y,&x);
      printf("%d,%d\n",x,y);
   }
   return 0;
}
void f1(int y, int *x) {
   y=y+*x;
   *x=*x+y;
   printf("%d,%d\n",*x,y);
}
```

(4 marks)

Note: Question No. 1 continues on Page 2

(b) The following program copies the contents of character string `a` into character string `b`. For every three characters copied from `a` to `b`, a character `'#'` is inserted into `b`. For example, if `a` is `"abcdefg"`, then `b` will be `"abc#def#g"` after executing the program. Write the missing code of the program in your answer book.

```
#include <stdio.h>
int main() {
   char a[80],b[80],*p;
   int i,k=0;
   p=a; gets(p);
   while ( *p!='\0' ) {
     /* missing code */
   }
   puts(b);
   return 0;
}
```

(5 marks)

(c) The following program reads an English sentence into `s`, and finds the length of the longest word in the sentence. For example, if the sentence is `"I am happy."`, then the length of the longest word `"happy"` in the sentence will be 5. Assume that each word is a sequence of English letters. Write the missing code of the program in your answer book.

```
#include <stdio.h>
int main() {
   char *p, s[80];
   int max=0,len=0;
   gets(s);
   p=s;
   while ( *p!='\0' ) {
      while ( /* missing code (i) */ ) {
         len++; p++;
      }
      if ( /* missing code (ii) */ )
         /* missing code (iii) */
      len=0; p++;
   }
   printf("Longest length is: %d\n", max);
   return 0;
}
```

(5 marks)

Note: Question No. 1 continues on Page 3

(d) The following program computes special numbers up to 1000. A special number is a 3 digit positive integer, in which the sum of the cubes of each digit is equal to the number. For example, the number 407 is a special number as 407 = 4×4×4 + 0×0×0 + 7×7×7. Write the missing code of the program in your answer book.

```
#include <stdio.h>
int main() {
   int x,y,z,a[10],num,i=0;
   for ( /* missing code (i) */ ) {
      x=num/100;
      /* missing code (ii) */
      if (x*100+y*10+z == x*x*x+y*y*y+z*z*z) {
         /* missing code (iii) */
      }
   }
   printf("Special numbers are: ");
   for (x=0; x<i; x++)
      printf("%d ",a[x]);
   return 0;
}
```

(5 marks)

(e) The following program finds the minimum of the maximum numbers of each row of a 2-dimensional array `a`. For example, if `a` is `{{1,3,5,2},{2,4,6,8},{8,6,4,9},{7,4,3,2}}`, then the maximum numbers will be 5, 8, 9 and 7 for rows 0, 1, 2 and 3 respectively, and the minimum of the maximum numbers will be 5. Write the missing code of the program in your answer book.

```
#include <stdio.h>
int main() {
   int a[4][4];
   int row,col,max,min;
   for (row=0; row<4; row++)
      for (col=0; col<4; col++)
         scanf("%d", &a[row][col]);
   for (row=0; row<4; row++) {
      /* missing code */
   }
   printf("Minimum is: %d\n", min);
   return 0;
}
```

(6 marks)

2. (a) What is the output of the following code?

```
#include <stdio.h>
void f2(int n, int *p);
int main() {
   int s;
   f2(5,&s);
   printf("Result: %d\n",s);
   return 0;
}
void f2(int n, int *p) {
   int a,b;
   if (n==1 || n==2)  *p=1;
   else {
      f2(n-1,&a); f2(n-2,&b);
      *p=a-b;
      printf("f2(): %d\n", *p);
   }
}
```

(5 marks)

(b) The following program reads the name and age of three persons into an array `man`, finds the person whose age is the middle one of the three persons, and prints the name and age of that person. For example, if `man` is `{{"Tom",18},{"John",19},{"Jim",20}}`, then `John` and `19` will be printed after executing the program. Write the missing code of the program in your answer book.

```
#include <stdio.h>
int main() {
   struct {
      char name[20]; int age;
   } man[3];
   int i,max,min;
   /* missing code */
   return 0;
}
```

(5 marks)

(c) The following **recursive** function `rStrLen()` accepts a character string `s` as parameter, and returns the length of the string. For example, if `s` is `"abcde"`, then the function `rStrLen()` will return 5. Write the missing code of the function in your answer book.

Note: Question No. 2 continues on Page 5

```
int rStrLen(char *s) {
   if ( /* missing code (i) */ )
      return /* missing code (ii) */
   else
      return /* missing code (iii) */
}
```

(4 marks)

(d) The following function `encode()` accepts two character strings `s` and `t` as parameters, encodes the characters in `s` to `t`, and passes the encoded string `t` to the caller via call by reference. During the encoding process, each `source` character is converted into the corresponding `code` character based on the following rules: `'a'→'d'; 'b'→'z'; 'z'→'a';` and `'d'→'b'`. For other source characters, the `code` will be the same as the `source`. For example, if the character string `s` is `"abort"`, then the encoded string `t` will be `"dzort"`. Write the missing code of the function in your answer book.

```
void encode(char *s, char *t) {
   typedef struct {
      char source; char code;
   } Rule;
   Rule table[] = { 'a','d', 'b','z',
          'z','a', 'd','b', '\0','\0' };
   Rule *p;
   while ( *s!='\0' ) {
      for (p=table; (*s != p->source) &&
          (p->source != '\0'); p++);
      /* missing code (i) */
   }
   /* missing code (ii) */
}
```

(5 marks)

(e) The following program reads the height of a tree pattern into `h`, and prints the pattern with `h` lines. For example, Figure Q2 shows the tree pattern to be printed when the user enters 5. The first line of the tree pattern consists of one asterisk; the second line consists of three asterisks centred below the one on the first line; the third line consists of five asterisks centred below those on the second line; and so forth. The **recursive** function `prtLine()` accepts a character `c` and an integer `n` as parameters, and prints the character `c` for `n` times. Write the missing code of the program in your answer book.

Note: Question No. 2 continues on Page 6

```
    *
   ***
  *****
 *******
*********
```

Figure Q2

```
#include <stdio.h>
void prtLine(char c, int n);
int main() {
   int i,h;
   scanf("%d", &h);
   for (i=1; i<=h; i++) {
      /* missing code (i) */
   }
   return 0;
}
void prtLine(char c, int n) {
   /* missing code (ii) */
}
```

(6 marks)

3. (a) The memory allocation process may be classified as static or dynamic.

(i) What is dynamic memory allocation? Explain the importance of the malloc() and free() functions.

(6 marks)

(ii) What is the difference between storing data on the heap versus on the stack?

(3 marks)

(b) Stacks and queues are special kinds of ordered lists in which insertions and deletions are restricted to only some specific positions.

(i) Briefly explain why recursive procedures are implemented using the stack data structure.

(3 marks)

Note: Question No. 3 continues on Page 7

(ii) What are the advantages of using circular queues over static queues?

(3 marks)

(iii) What are the differences between a linked list and a queue?

(2 marks)

(c) Write the missing code for the function `sortedInsert()` in your answer book. The `sortedInsert()` function, when given a list that is sorted in increasing order and a single node, inserts the node into the correct sorted position in the list.

```
1  typedef struct _listnode {
2     int item;
3     struct _listnode *next;
4  } ListNode;
5
6  void sortedInsert(ListNode **headPtr, ListNode
   *newNode) {
7
8     if (*headPtr == NULL || (*headPtr)->item >=
      newNode->item) {
9        /* missing code (i) */
10    }
11    else {
12       ListNode *current = *headPtr;
13       while (current->next != NULL
         && /* missing code (ii) */ ) {
14          /* missing code (iii) */
15       }
16
17       /* missing code (iv) */
18    }
19 }
```

(8 marks)

4. (a) A binary tree is a tree which is either empty or has at most two subtrees, with each of the subtrees also being a binary tree.

(i) What is the maximum number of nodes possible in a binary tree of depth d?

(3 marks)

Note: Question No. 4 continues on Page 8

(ii) A binary tree has 10 nodes. The inorder and preorder traversals of the binary tree produce the following sequence of nodes:
Inorder : D B H E A I F J C G
Preorder : A B D E H C F I J G

Draw the binary tree and write the order of nodes visited using post-order traversal.

(5 marks)

(b) Write the missing code for the function `levelOrderTraversal()` in your answer book. As its argument, the function takes a pointer to the root node of a binary tree. The **iterative** function `levelOrderTraversal()` prints a level-by-level traversal of the binary tree using a queue, starting at the root node level.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct _treeNode {
5     struct _treeNode *leftPtr;
6     int data;
7     struct _treeNode *rightPtr;
8  } TreeNode;
9
10 typedef struct _queueNode {
11    TreeNode *data;
12    struct _queueNode *nextPtr;
13 } QueueNode;
14
15 void levelOrderTraversal(TreeNode *treePtr);
16 void insertNode(TreeNode **treePtr, int value);
17 void enqueue(QueueNode **headPtr, QueueNode
   **tailPtr,TreeNode *node);
18 TreeNode* dequeue(QueueNode **headPtr,
   QueueNode **tailPtr);
19 int isEmpty(QueueNode *headPtr);
20
21 void levelOrderTraversal(TreeNode **ptr) {
22    QueueNode *tail = NULL;
23    QueueNode *head = NULL;
24    TreeNode  *node = NULL;
25
26    if (ptr != NULL) {
27       enqueue(&head, &tail, ptr);
```

Note: Question No. 4 continues on Page 9

```
28       while (!isEmpty(head)) {
29          /* missing code (i) */
30          if (node->leftPtr != NULL) {
31             /* missing code (ii) */
32          }
33          if (node->rightPtr != NULL) {
34             /* missing code (iii) */
35          }
36       }
37    }
38 }
```

(9 marks)

(c) Write the missing code for the function `binaryTreeSearch()` in your answer book. As arguments, the function takes a pointer to the root node of a binary tree and a search key to be located. If a node which contains the search key is found, the function returns a pointer to that node; otherwise, the function returns a `NULL` pointer.

```
1  typedef struct _treeNode {
2     struct _treeNode *leftPtr;
3     int data;
4     struct _treeNode *rightPtr;
5  } TreeNode;
6
7  TreeNode *binaryTreeSearch(TreeNode *treePtr,
   const int key) {
8     if (treePtr == NULL) {
9        return NULL;
10    }
11    else if (treePtr->data == key) {
12       return treePtr;
13    }
14    else if (/* missing code (i) */) {
15       return /* missing code (ii) */
16    }
17    else if (/* missing code (iii) */) {
18       return /* missing code (iv) */
19    }
20   return NULL;
21 }
```

(8 marks)

END OF PAPER

CE1007 DATA STRUCTURES
CZ1007 DATA STRUCTURES

Please read the following instructions carefully:

1. **Please do not turn over the question paper until you are told to do so. Disciplinary action may be taken against you if you do so.**

2. You are not allowed to leave the examination hall unless accompanied by an invigilator. You may raise your hand if you need to communicate with the invigilator.

3. Please write your Matriculation Number on the front of the answer book.

4. Please indicate clearly in the answer book (at the appropriate place) if you are continuing the answer to a question elsewhere in the book.