NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2014-2015

CE1007/CZ1007 – DATA STRUCTURES

Nov/Dec 2014 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 6 pages.
2. Answer **ALL** questions.
3. This is a closed-book examination.
4. All questions carry equal marks.

---

1. (a) State the situations where a programmer should use the call by reference mode of parameter passing when implementing a function.

(4 marks)

(b) Remove the errors from the following function by rewriting the corrected function in your answer book. Circle the changes that you made to the original function.

```
/* This function reverses the characters in a
string */
void reverse(char *s){
   int n;
   char *tmp, *p, *q;
   n = strlen(s);
   q = (n>0) ? (s+n) : s;
   for (p=s; p<q; ++p, --q){
      *tmp = *p;
      *p = *q;
      *q = *tmp;
   }
}
```

(8 marks)

Note: Question No. 1 continues on Page 2

(c) The following program maintains a database of 100 employee records. For each employee record, it should contain the names (last name and first name, each of at most 20 characters), age, gender ('M' or 'F') and salary. In the program, it first declares an appropriate structure for an employee record. The function `readEmployee()` reads and returns an employee record to the caller via the parameter `emp`. The function `printEmployee()` takes an array of employee records `emp[SIZE]` as parameter and prints each employee record's information stored in the array. Write the three missing code segments `(i)`, `(ii)` and `(iii)` in the program.

```
#include <stdio.h>
#define SIZE 100
struct employee
{
   /* (i) missing code segment */
};
void readEmployee(struct employee *emp);
void printEmployee(struct employee emp[SIZE]);
int main()
{
   struct employee e[SIZE];
   int i;
   printf("Enter %d records: ", SIZE);
   for (i=0; i<SIZE; i++) {
      readEmployee(&e[i]);
   }
   printEmployee(e);
   return 0;
}
void  readEmployee(struct employee *emp)
{
   /* (ii) missing code segment */
}
void  printEmployee(struct employee emp[SIZE])
{
   /* (iii) missing code segment */
}
```

(13 marks)

2. In the following program, the functions `allEven1()` and `allEven2()` take one integer argument and return either 1 or 0, according to whether or not all the digits of the argument are even. For example, if `number` is `2468`, the function will return 1; and if `number` is `12345`, the function will return 0. In the program, an iterative approach is used for the implementation of `allEven1()`, and a recursive approach is used for the implementation of `allEven2()`. In `allEven3()`, it uses a recursive approach for the implementation and returns the result indirectly via the parameter `result`, instead of the result being returned directly.

```
#include <stdio.h>
int allEven1(int number);
int allEven2(int number);
void allEven3(int number, int *result);
int main()
{
   int number, result;
   printf("Enter a number: ");
   scanf("%d", &number);
   printf("allEven1(): %d\n", allEven1(number));
   printf("allEven2(): %d\n", allEven2(number));
   allEven3(number, &result);
   printf("allEven3(): %d\n", result);
   return 0;
}
int allEven1(int number)
{
   /* (i) missing code segment */
}
int allEven2(int number)
{
   /* (ii) missing code segment */
}
void allEven3(int number, int *result)
{
   /* (iii) missing code segment */
}
```

(a) Write the three missing code segments `(i)`, `(ii)` and `(iii)` in the program.

(21 marks)

(b) Compare and contrast the iterative approach and recursive approach as used in your respective implementation of the functions `allEven1()` and `allEven2()` in Q2(a).

(4 marks)

3. (a) Describe two advantages of linked list when compared to array.

(4 marks)

(b) A stack stores integers in ascending order (see stack1 in Figure Q3). Your task is to insert a new integer 7 into stack1 while maintaining the order (see the resulting stack1 shown in Figure Q3). Stack2 has been initialized as an empty stack and may be used in your code. Suppose the following two functions have been defined:

```
int  pop(stack *s)
void push(int i, stack *s)
```

For example, `push(pop(s1),s2)` will pop the integer 15 from stack1 and push it into stack2, where `s1` and `s2` represent pointers to stack1 and stack2 respectively. Write down the operation sequence that can insert integer 7 into stack1 correctly. The operation sequence should only contain the operations of `pop()` and `push()` defined above with variables `s1` and `s2`, and the constant integer `7`.

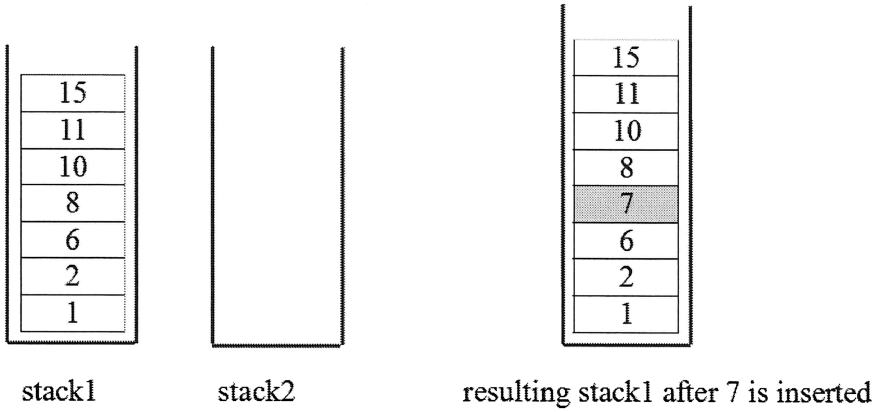


Figure Q3

(8 marks)

Note: Question No. 3 continues on Page 5

(c) A linked list is used to store the names of members of a club. People can join or quit the club any time. Write the functions to insert and remove members. The function `add(ListNode *head, char *s)` adds a new member name `s` into the linked list (note that no special order of names is required in the linked list), where `head` is a pointer that points to the first node in the linked list. The function `remove(ListNode *head, char *s)` deletes a member name `s` from the linked list. If the name is not found in the linked list, the function will display the message "no such name is found!" on the screen. You may assume that the linked list is always non-empty (i.e., `head!=NULL`). The function prototypes are given below:

```
void add(ListNode *head, char *s);
void remove(ListNode *head, char *s);
```

`ListNode` is defined as follows:

```
typedef struct _listnode{
     char name[20];
     _listnode *next;
} ListNode;
```

(13 marks)

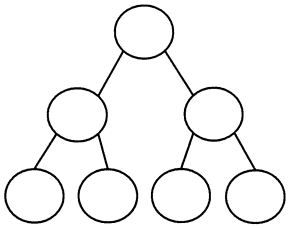
4. (a) Draw a binary search tree that may be generated from the integer set {1,2,3,4,5,6,7}. Describe the concept of a *binary search tree*.

(5 marks)

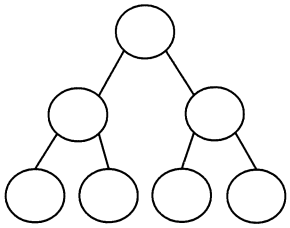
(b) Three different binary trees with missing node data are shown in Figure Q4. Each node in the tree stores a character, which belongs to the set C = {'*a*', '*b*', '*c*', '*d*', '*e*', '*f*', '*g*'}. Draw these three trees and choose a proper character selected from the set C for every node, so that the output sequences of these three trees (visited using Pre-order, In-order and Post-order traversals respectively) are the same (i.e., "*a b c d e f g*"). You may assume that the character stored in a node is printed when the node is visited by the traversal.

(10 marks)

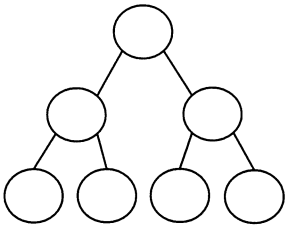
Note: Question No. 4 continues on Page 6



(i) A binary search tree visited by Pre-order traversal.



(ii) A binary search tree visited by In-order traversal.



(iii) A binary search tree visited by Post-order traversal.

Figure Q4

(c) Write a C function `printSmallerValues()` that accepts a pointer to the root node of a binary search tree and prints all integers stored in the tree that are smaller than a given value `m`. The function prototype is given as follows:

```
void printSmallerValues(BTNode *node, int m);
```

A `BTNode` structure is defined as follows:

```
typedef struct _btnode{
    int item;
    struct _btnode *left;
    struct _btnode *right;
} BTNode;
```

(10 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.