NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2015-2016

CE1007/CZ1007 – DATA STRUCTURES

Nov/Dec 2015 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 are the outputs of the following code?

(i)
```
#include <stdio.h>
int main() {
   char str[]="ABCCDA", c;
   int k;
   for (k=0; (c=str[k])!='\0'; k++) {
      switch(c) {
         case 'A': putchar('%');
            continue;
         case 'B': ++k;
            break;
         default: putchar('*');
         case 'C': putchar('&');
            break;
      }
      printf("\n");
   }
   putchar('#');
   return 0;
}
```

(4 marks)

Note: Question No. 1 continues on Page 2

(ii)
```
#include <stdio.h>
int main() {
   char *a="PROGRAM";
   char b[]="program";
   int i=0;
   printf("%c%s\n",*a,b+1);
   while (*(a+i)) { putchar(*(a+i)); i++; }
   printf("\n");
   while (--i)
      putchar(*(b+i));
   printf("\n");
   printf("%s\n",&b[3]);
   return 0;
}
```

(4 marks)

(b) Write the missing code of the following programs in your answer book.

(i)
```
/* The following program merges two alphabetically ordered
character strings a and b into character string c according to
alphabetical order. For example, if a is "agikmpq" and b is
"bcdefhjlnr", then the string c will be "abcdefghijklmnpqr". */
#include <stdio.h>
#include <string.h>
int main() {
   char a[]="agikmpq";
   char b[]="bcdefhjlnr";
   char c[80],*p;
   int i=0,j=0,k=0;
   while (a[i]!='\0' && b[j]!='\0') {
      if (a[i]<b[j])
      { /* missing code (i) */ }
      else
      { /* missing code (ii) */ }
      k++;
   }
   c[k]='\0';
   if ( /* missing code (iii) */ )
      p=b+j;
   else
      p=a+i;
   strcat(c,p);
   puts(c);
   return 0;
}
```

(6 marks)

Note: Question No. 1 continues on Page 3

(ii) /* In the following program, the function **compare()** compares two character strings **s** and **t** according to alphabetical order. If **s** is greater than **t**, then it will return a positive value. Otherwise, it will return a negative value. For example, if **s** is "boy" and **b** is "girl", then the function will return -5 which is the difference between the ASCII values of 'b' and 'g'. If **s** is "car" and **t** is "apple", then it will return 2 which is the difference between the ASCII values of 'c' and 'a'. */

```
#include <stdio.h>
int compare(char *s, char *t);
int main() {
   char a[80],b[80];
   gets(a); gets(b);
   printf("%d",compare(a,b));
   return 0;
}
int compare(char *s, char *t) {
   for ( ; *s==*t; /* missing code (i) */ )
      if (*s=='\0')
         return 0;
   return ( /* missing code (ii) */ );
}
```

(4 marks)

(iii) /* In the following program, the function **countStr()** counts the number of times that the substring **substr** appears in the character string **str**. If the **substr** is not contained in **str**, then it will return 0. Note that you should not declare any other variables in the function. */

```
#include <stdio.h>
int countStr(char str[], char substr[]);
int main() {
   char str[80],substr[80];
   gets(str); gets(substr);
   printf("%d\n", countStr(str, substr));
   return 0;
}
int countStr(char str[], char substr[]) {
   int i,j,k,count=0;

   /* missing code */

   return count;
}
```

(7 marks)

2. (a) What are the outputs of the following code?

(i)
```
#include <stdio.h>
int main() {
   int a[10],b[10],*pa,*pb,i;
   pa=a; pb=b;
   for (i=0; i<3; i++,pa++,pb++) {
      *pa=i;
      *pb=2*i;
      printf("%d %d ",*pa,*pb);
   }
   printf("\n");
   pa=&a[0]; pb=&b[0];
   for (i=0; i<3; i++) {
      *pa=*pa+i; *pb=*pb+i;
      printf("%d %d\n", *pa++, *pb++);
   }
   return 0;
}
```

(4 marks)

(ii)
```
#include <stdio.h>
int main(){
   typedef struct {
      int x;
      int *y;
   } data;
   int a[]={ 10,20,30,40,50 };
   data b[5]={ 1,&a[0],2,&a[1],3,&a[2],
               4,&a[3],5,&a[4] };
   data *p;
   p=b;
   printf("%d\n",(++p)->x);
   printf("%d\n",*(++p)->y);
   printf("%d\n",*p++->y);
   printf("%d\n",++(*(++p)->y));
   return 0;
}
```

(4 marks)

Note: Question No. 2 continues on Page 5

(b) Find the errors in the following code. Correct the errors by stating the line numbers and rewriting the corrected statements in your answer book.

/* The following program processes an array of student records. For each student record, it stores the student id and name. In the program, it reads in each student's information, and then prints the student information on the display. The functions **`input()`** and **`output()`** are used for the reading and printing of student information. */

```
 1 #include <stdio.h>
 2 #define SIZE 10
 3 struct Stud {
 4    int id;
 5    char name[10];
 6 };
 7 void input(struct Stud s);
 8 void output(struct Stud s);
 9 int main() {
10    struct Stud s[SIZE];
11    int i;
12    for (i=0; i<SIZE; i++)
13       input(s[i]);
14    for (i=0; i<SIZE; i++)
15       output(s[i]);
16    printf("\n");
17    return 0;
18 }
19 void input(struct Stud s) {
20    printf("Student ID: ");
21    scanf("%d", &s.id);
22    printf("Student Name: ");
23    scanf("%s", s.name);
24 }
25 void output(struct Stud s) {
26    printf("%s(%d) ", s.name, s.id);
27 }
```

(5 marks)

Note: Question No. 2 continues on Page 6

(c) Write the missing code of the following programs in your answer book.

(i) /* The following program reads 5 integer numbers into an array **a**, finds the index positions of the largest number and smallest number, swaps these two numbers, and prints their original index positions and the resultant array of numbers to the display. For example, if the input numbers for **a** are 1, 2, 3, 4 and 5, then the resultant array **a** will be 5, 2, 3, 4 and 1 when it is printed. */

```
#include <stdio.h>
int main() {
   int a[5],max,min,i=0,j=0,k;
   for (i=0; i<5; i++) scanf("%d",a+i);
   min = *a;
   for (i=1;i<5;i++)
      if (*(a+i)<min) {/*missing code (i)*/}
   max = *a;
   for (i=1; i<5; i++)
      if (*(a+i)>max) {/*missing code (ii)*/}
   /* missing code (iii) */
   printf("Max position = %d\n",k);
   printf("Min position = %d\n",j);
   for (i=0; i<5; i++) printf("%d ",*(a+i));
   return 0;
}
```

(5 marks)

(ii) /* In the following program, the **recursive** function **findmax()** finds the index position of the maximum number in an array of integer numbers. Note that you should not declare any other variables in the function. */

```
#include <stdio.h>
#define SIZE 10
void findmax(int *a,int n,int i,int *index);
int main() {
   int a[SIZE],i,index=0;
   printf("Enter %d data: ", SIZE);
   for (i=0; i<SIZE; i++) scanf("%d",a+i);
   findmax(a,SIZE,0,&index);
   printf("Maximum number = %d\n", a[index]);
   printf("Index position = %d\n", index);
   return 0;
}
void findmax(int *a,int n,int i,int *index){
   if (i<n) {
     /* missing code */
   }
}
```

(7 marks)

3. (a) A linked list is a data structure consisting of a number of nodes chained together to represent a sequence.

(i) What are the differences between a linked list and a stack?

(2 marks)

(ii) What are the advantages of using linked lists over arrays?

(2 marks)

(iii) Describe briefly the four types of linked lists.

(4 marks)

(b) Assume that a queue is maintained by a circular array QUEUE with N = 12 where N is the size of the QUEUE. Find the number of elements in the QUEUE if

(i) Front = 8, Rear = 4.

(2 marks)

(ii) Front = 6, Rear = 7 and then two elements are deleted.

(3 marks)

(c) Write the missing code of the function **`reverse()`** in your answer book. The **iterative** function **`reverse()`** reverses the links in a linked list such that the last node becomes the first, and the first node becomes the last by traversing the linked list only once. Note that `**head` is a reference (pointer to pointer) to the head of the linked list.

```
typedef struct _listnode {
   int item;
   struct _listnode *next;
} ListNode;

void reverse(ListNode **head)
{

   /* missing code */

}
```

(12 marks)

4. (a) Write the missing code of the function **preOrderIterative()** in your answer book. The **iterative** function **preOrderIterative()** prints the pre-order traversal of a binary tree using a stack.

```
#include <stdio.h>
#define MAX_SIZE 100

struct Node {
   int data;
   struct Node *left, *right;
};

struct Stack {
   int size;
   int top;
   struct Node **array;
};

struct Stack *createStack(int size);

int isEmpty(struct Stack *s);

void push(struct Stack *s, struct Node *n);

struct Node *pop(struct Stack *s);

void preOrderIterative(struct Node *root) {
   struct Stack *stack;
   struct Node *node;

   if (root == NULL)  return;
   stack = createStack(MAX_SIZE);
   /* missing code (i) */
   while (!isEmpty(stack)) {
      node = /* missing code (ii) */
      printf("%d ", node->data);
      if (node->right)
         /* missing code (iii) */
      if (node->left)
         /* missing code (iv) */
   }
}
```

(9 marks)

Note: Question No. 4 continues on Page 9

(b) Consider the binary search tree in Figure Q4 and answer the following questions.

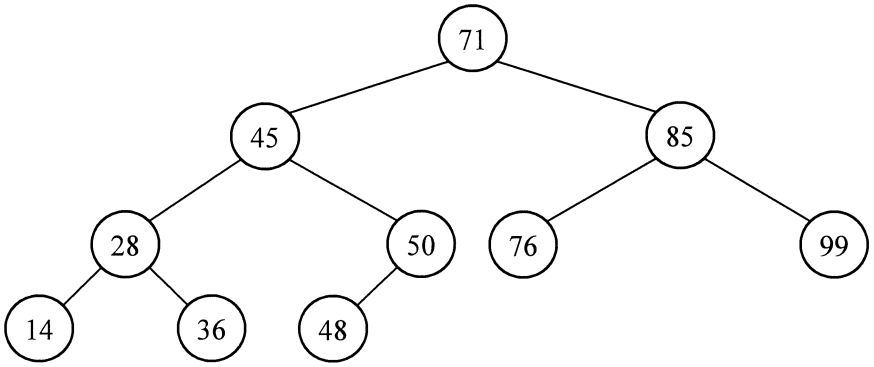

Figure Q4

(i) Write the order of nodes visited using the pre-order traversal?

(2 marks)

(ii) If node 45 is removed from the binary search tree, what would be the order of nodes visited using the post-order traversal?

(4 marks)

(c) Two binary trees are similar if they are both empty, or if they are both non-empty and the left and right subtrees are similar. Write a **recursive** C function **`similar()`** to determine whether the two binary trees, tree1 and tree2, are similar. This function returns 1 if the two binary trees are similar; otherwise it returns 0. The function prototype is given as follows:

```
int similar(BTNode *tree1, BTNode *tree2);
```

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.