NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 2 EXAMINATION 2011-2012

CE1007/CZ1007 – DATA STRUCTURES

April/May 2012 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 5 pages.
2. Answer **ALL** questions.
3. This is a closed-book examination.
4. All questions carry equal marks.

---

1. (a) State the output produced by the `printf()` statements of the program in Figure Q1.

```
#include <stdio.h>
void f(int x, int *y);
int main( )
{
   int h=8, k=0;
   f(h, &k);
   printf("h = %d, k = %d\n", h, k);
   return 0;
}
void f(int x, int *y)
{
   printf("x = %d, *y = %d\n", x, *y);
   if (x > 0)  {
      *y += x;
      f(x-2, y);
   }
   printf("x = %d, *y = %d\n", x, *y);
}
```

Figure Q1

(11 marks)

Note: Question No. 1 continues on Page 2

(b) Give one advantage of using iteration instead of recursion and one advantage of recursion instead of iteration. Explain your answers.

(4 marks)

(c) Write a recursive C function *digitPos* that takes a positive integer argument *n* and returns the position of the first appearance of a specified *digit* in the integer *n*. The position of the digit is counted from the right and starts from 1. If the required digit is not in the number, the function should return 0. For example, *digitPos(42345, 4)* returns 2 and *digitPos(123, 6)* returns 0. No error checking on the parameters is required in the function. In addition, you must not use an **array** or a **string**. The prototype of the function is given below:

```
int digitPos(int n, int digit);
```

(10 marks)

2. (a) What are the differences and similarities between an array and a string in C programming?

(3 marks)

(b) Write a C function *findString* that takes two character string arguments, *s* and *t* as input and returns 1 if *s* is a substring of *t* (i.e. if *s* is contained in *t*) and -1 if not. For example, the function will return 1 if *s* is "123" and *t* is "abc123xyz", but it will return -1 if *s* is "123" and *t* is "abc12xyz". You must not use any of the string functions from the standard C libraries. The prototype of the function is given below:

```
void findString(char *s, char *t);
```

(10 marks)

(c) Determine the output of the program given in Figure Q2, where the addresses of `ch`, `ptr1`, `ptr2` and `ptr3` are 0022FF0B, 0022FEE4, 0022FEE0 and 0022FEDC respectively.

Note: Figure Q2 is on Page 3

```
#include <stdio.h>
int main( )
{
   char ch;
   char ar1[4][6] = {
                    {'P', 'e', 't', 'e', 'r'},
                    {'J', 'o', 'h', 'n', 'n', 'y'},
                    {'M', 'a', 'r', 'y'},
                    {'K', 'e', 'n', 'n', 'y'}
                     };
   char *ar2;
   char *ar3[2];
   char *ptr1, **ptr2, ***ptr3;

   ch = 'a';
   ptr1 = &ch; ptr2 = &ptr1; ptr3 = &ptr2;
   printf("Output1 = %p\n", &ptr2);
   printf("Output2 = %p\n", **ptr3);
   printf("Output3 = %p\n", *ptr3);
   printf("Output4 = %p\n", *ptr2);

   ar2 = &ar1[1][1];
   ar3[0] = &ar1[0][0];
   ar3[1] = &ar1[1][0];
   printf("Output5 = %s\n", &ar1[1][4]);
   printf("Output6 = %s\n", ar2-3);
   printf("Output7 = %s\n", ar3[0]+1);
   printf("Output8 = %s\n", ar3[1]+2);
   return 0;
}
```

Figure Q2

(12 marks)

3. (a) Explain the importance of `malloc()` and `free()` in the implementation of dynamic data structures in C programming.

(3 marks)

(b) Consider the code given in Figure Q3 for the C structure defining a node in a singly-linked list of integers.

Note: Question No. 3 continues on Page 4

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

Figure Q3

Modify the `ListNode` structure to support the following requirements:

I. The linked list should be able to store any form of data in its nodes instead of just integer values.

II. Given a data item *"X"* and an integer *P*, the linked list should support efficient insertion of a copy of the node *N* containing item *"X"*, in the position *P* nodes before *X*. You may assume that the new position exists and *"X"* can be found.

For example, if the string *"Test Item"* is the data item in the seventh node, and *P = 3*, a copy of the seventh node *N* should be inserted in the fourth position of the linked list, and *N* will now become the eighth node.

(7 marks)

(c) A bank has asked you to implement an application for keeping track of customer records and banking activities. While the application is running, customer records should be maintained in ascending alphabetically sorted order (A-Z). Each customer may have any number of accounts with the bank; the bank provides both checking and savings accounts. Each account should maintain a record of all transactions carried out since the opening of that account.

When designing your application, you should consider that this bank is the only financial institution in its area. This means that its customer list rarely changes and customers infrequently sign up for new accounts, although many transactions are made every day.

Using appropriate diagrams, state and justify the data structure (or combinations of data structures) used in your application.

(15 marks)

4. (a) Draw the Binary Search Tree generated after inserting the following numbers in the order that they appear:

99, 4, 19, 36, 64, 30, 3, 1

(6 marks)

(b) Does the tree you generated in Q4(a) allow for efficient search? Explain your answer.

(7 marks)

(c) Write the pseudocode for a function *IsBST* which checks whether a given Binary Tree `B` is also a Binary Search Tree.

(12 marks)

END OF PAPER

NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2012-2013

CE1007/CZ1007 – DATA STRUCTURES

Nov/Dec 2012 Time Allowed: 2 hours

INSTRUCTIONS

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

---

1. (a) Write a C function `minMax()` that takes a 5x5 two-dimensional array of integers *a* as a parameter. The function returns the minimum and maximum numbers of the array back to the caller through the two parameters *min* and *max* respectively. The function prototype is given below:

```
void minMax(int a[5][5], int *min, int *max);
```

(8 marks)

(b) Write a recursive C function `sumOddDigits()` that takes a positive integer parameter *n* and returns the sum of the odd digits of the integer to the caller. For example, `sumOddDigits(12345)` returns 9. The prototype of the function is given below:

```
int sumOddDigits(int n);
```

(8 marks)

(c) Modify your answer in Q1(b) above so that instead of the result being returned directly, it is returned indirectly, via a second parameter.

(5 marks)

Note: Question No. 1 continues on Page 2

(d) Compare and contrast the two methods of parameter passing used in Q1(b) and Q1(c).

(4 marks)

2. (a) The function `strrchr()` has the following function prototype:

```
char *strrchr(char *s, char ch);
```

This C function locates the last occurrence of *ch* in the string pointed to by *s*. The function returns a pointer to the character, or a null pointer if *ch* does not occur in the string. Write the code for the function without using any of the standard library string functions.

(8 marks)

(b) You are required to write a program to maintain bank account records. For each bank customer, the following information is required:

- customer name (first_name and last_name, each of at most 10 characters);
- address (at most 80 characters);
- account number (an integer);
- account balance (a real number); and
- account opening date (day, month and year).

(i) Declare an appropriate structure called *account* to represent a bank account record with the customer information given above.

(5 marks)

(ii) Write a C program that repeatedly reads in customer data from the user and prints the customer data on the screen until the customer name *End Customer* (i.e., first_name last_name) is read. Your program should include the following two functions: the function `nextCustomer()` reads and returns a record for a single customer to the caller via a parameter *acct*, and the function `printCustomer()` takes a parameter *acct* and then prints the customer information. The prototypes of the two functions are given below:

```
void nextCustomer(struct account *acct);
void printCustomer(struct account acct);
```

(12 marks)

3. (a) Describe what happens when a long-running C program that is continually expanding its data structures does not free unused memory that was dynamically allocated using `malloc()`.

(4 marks)

(b) A stack may be implemented using an array or a linked list. State one advantage and one disadvantage of each approach.

(6 marks)

(c) Write a function `copyEvenItems()` that accepts a linked list of integers and returns a new linked list made up of duplicates of only the even-valued items. For example, calling `copyEvenItems()` on the list `[3, 3, 2, 1, 4, 6, 7, 9]` will return a new list `[2, 2, 4, 4, 6, 6]`. You may assume that the input linked list contains only positive numbers. You may also make use of the get(), find() and insert() functions for the LinkedList data structure. The prototype of the function is given below:

```
linkedlist * copyEvenItems(linkedlist *ll)
```

(12 marks)

(d) Describe a situation where storing items in an array is clearly better than storing them in a linked list.

(3 marks)

4. (a) Consider the binary tree **T** in Figure Q4.

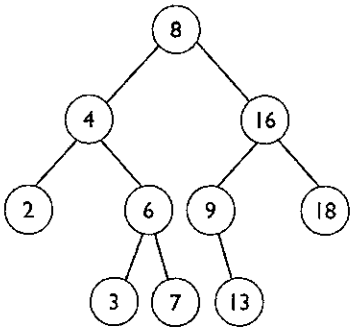


Figure Q4

Note: Question No. 4 continues on Page 4

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

(3 marks)

(ii) Write the order of nodes visited using a post-order traversal.

(3 marks)

(iii) Is the binary tree **T** a binary search tree? Explain.

(3 marks)

(b) Write a recursive C function `sumOddNodes()` that accepts a pointer to the root node of a binary tree of integers and returns the sum of all odd numbers stored in the tree. The prototype of the function is given below:

```
int sumOddNodes(btnode *node);
```

A btnode structure is defined as follows:

```
typedef struct btnode{
     int item;
     struct btnode *left;
     struct btnode *right;
} btnode;
```

(10 marks)

(c) If the pre-order and in-order traversal of a binary tree produce the same sequence, what can be said about the structure of the tree?

(6 marks)

END OF PAPER

NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 2 EXAMINATION 2012-2013

CE1007/CZ1007 – DATA STRUCTURES

Apr/May 2013 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 4 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>
   int f(int n);
   int main()
   {
      printf("Result = %d\n", f(6));
      return 0;
   }
   int f(int n)
   {
      int a, b;
      if (n>2){
         a = f(n-1);
         b = f(n-2);
         printf("a = %d, b = %d\n", a, b);
         return a+b;
      }
      else
         return 2;
   }
```

(8 marks)

Note: Question No. 1 continues on Page 2

(b) Given the following C declarations:

```
struct student {
    int age;
    char *name;
};
struct student b[3] = {
    18, "Peter", 19, "Mary", 20, "John"
};
struct student *p = b;
```

Assume that the address of the first element of the array `b[3]` is 1010 and the following operations are executed sequentially, in order, from (i) to (vi). What are the values of the following expressions?

(i) `++p->age`
(ii) `(*p).name`
(iii) `*p->name-1`
(iv) `*++p->name`
(v) `*p++->name`
(vi) `*(++p)->name`

(7 marks)

(c) Write a **recursive** C function `reverseAr()` that reverses the contents of an array. The function takes in two arguments `ar[]` and `n`, which are an array of characters and an integer specifying the size of the array respectively. For example, if the array contains, in order, 1, 2, 3, 4, 5, then after reversal, its content should be, in order, 5, 4, 3, 2, 1. The code should not use any other arrays. The function prototype is given as follows:

```
void reverseAr(char ar[], int n);
```

(10 marks)

2. (a) Write a C function `squeeze()` that removes any occurrences of a given character `c` from the string `str`. For example, if the string content is `"abcdabcd"` and the character is `'b'`, then after removal, its content should be `"acdacd"`. The code should not use any of the standard string library functions. The function prototype is given as follows:

```
void squeeze(char str[], char c);
```

(7 marks)

Note: Question No. 2 continues on Page 3

(b) The C function `strcmp()` compares the string pointed to by `s1` to the string pointed to by `s2`. If the string pointed to by `s1` is greater than, equal to, or less than the string pointed to by `s2`, then it returns 1, 0 or –1 respectively. Write the code for the function without using any of the standard string library functions. The function prototype is given as follows:

```
int strcmp(char *s1, char *s2);
```

(9 marks)

(c) Given the following C declaration:

```
char *a[2][3] = {"abc", "defghi", "ijkl",
                 "mnopqr", "stuvm", "xyz"};
```

Assume that the address of the first element of the array is 1010 and the following operations are executed sequentially, in order, from (i) to (v). What are the values of the following expressions?

(i) `***(a+1)`
(ii) `**a[0]`
(iii) `(*(*(a+1)+1))[4]`
(iv) `*(a[1][2]+2)`
(v) `**(a+1)`

(9 marks)

3. (a) Explain how a memory leak occurs in C programs and why it is undesirable.

(6 marks)

(b) Write a C function `exchange()` that traverses a linked list of integers at most once, finds the nodes storing the maximum and minimum values in the list, and swaps those nodes. You may assume that all values stored in the list are unique. The function prototype is given as follows:

```
void exchange(linkedlist *ll);
```

For example, calling `exchange()` on the list [28 15 8 6 2 14] would result in the modified list [2 15 8 6 28 14].

(12 marks)

Note: Question No. 3 continues on Page 4

(c) A queue can be simulated using two stacks. Using appropriate diagrams and/or pseudocode, describe how the `enqueue()` and `dequeue()` functions would be implemented for this simulated queue.

(7 marks)

4. (a) State two differences between the linked list and binary tree data structures.

(6 marks)

(b) Draw all possible minimal-height Binary Search Trees that store the list *16, 44, 99, 1*.

(10 marks)

(c) Write a C function `countonechild()` that accepts a pointer to the root node of a binary tree and returns the number of nodes with exactly one child node. The function prototype is given as follows:

```
int countonechild(btnode *node);
```

(9 marks)

END OF PAPER

NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2013-2014

CE1007/CZ1007 – DATA STRUCTURES

Nov/Dec 2013 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 5 pages.
2. Answer **ALL** questions.
3. This is a closed-book examination.
4. All questions carry equal marks.

---

1. (a) Write a C function `compute()` that takes a 4x4 two-dimensional array `matrix` of floating point numbers as a parameter. The function computes the average of the first three elements of each row of the array and stores it at the last element of the row. For example, the contents of `matrix`, before and after calling the `compute()` function, are given below:

Before `compute()` : 4x4 matrix	After `compute()` : 4x4 matrix
1 2 3 0	1 2 3 2
4 5 6 0	4 5 6 5
7 8 9 0	7 8 9 8
3 4 5 0	3 4 5 4

The function prototype is given as follows:

```
void compute(float matrix[4][4]);
```

(8 marks)

Note: Question No. 1 continues on Page 2

(b) What does the function `f()` do in the following program? State the output of the program.

```
#include <stdio.h>
   int f(char *s1, char *s2);
   int main()
   {
      char a[80] = "abcdefg";
      char b[80] = "ace";
      int n;

      n = f(a,b);
      printf("%d %s", n, a);
      return 0;
   }
   int f(char *s1, char *s2)
   {
      int i=0,j;
      char *s = s1;

      while (*(s2+i) != '\0') {
         for (j=0, s1=s; *s1 != '\0'; s1++)
            if (*s1 != *(s2+i))  s[j++] = *s1;
         s[j] = '\0';
         i++;
      }
      return j;
   }
```

(7 marks)

(c) Write a C program that reads in five words separated by space, finds the first and last words according to ascending alphabetical order, and prints them on the screen. A sample input and output session is given below:

```
Enter 5 words: banana apple orange papaya kiwi
First word = apple, Last word = papaya
```

Note that the user input is underlined. Hint: you may use an array of strings to store the input words for subsequent processing.

(10 marks)

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

```
#include <stdio.h>
   int main()
   {
      struct s1 {
         char c[4], *s;
      } e1 = {"abc", "123"};
      struct s2 {
         char *p;
         struct s1 ss1;
      } e2 = {"xyz", {"def","456"}};
      printf("%c%c\n", e1.c[0], *e1.s);
      printf("%s%s\n", e1.c, e1.s);
      printf("%s%s\n", e2.p, e2.ss1.s);
      printf("%s%s\n", ++e2.p, ++e2.ss1.s);
      return 0;
   }
```

(7 marks)

(b) Given any positive integer n, the function `fun(n)` is defined based on the following mathematical formulae:

```
fun(n) = 1                 if n = 1
fun(n) = fun(n/2)          if n is even
fun(n) = 2*fun((n-1)/3)    if n>1 and n is odd
```

Write a recursive C function to implement `fun(n)`. The function computes and returns the result. The function prototype is given as follows:

```
int fun(int n);
```

(8 marks)

(c) Write a recursive C function `countZeros()` that counts the number of zeros in a specified positive number num. The function passes the result to the calling function through the parameter `count`. For example, `countZeros(105006)` returns 3 and `countZeros(1357)` returns 0. The function prototype is given as follows:

```
void countZeros(int num, int *count);
```

(10 marks)

3. (a) Explain the importance of the malloc() and free() functions when using dynamic data structures in C.

(6 marks)

(b) State two differences between a linked list and an array.

(4 marks)

(c) Describe an application and a dataset where a linked list is a more suitable data structure than an array.

(6 marks)

(d) Write a function `moveOddItemsToBack()` that accepts an ascending sorted linked list of integers and moves all odd values to the back of the linked list. The odd values should be maintained in ascending sorted order. The function should return the number of odd values that were encountered.

```
int moveOddItemsToBack(LinkedList *ll)
```

If you use any of the LinkedList functions discussed in class, you must explain any impact that they have on the efficiency of your function.

(9 marks)

4. (a) Consider an empty stack of integers. Let the numbers 1, 2, 3, 4, 5, 6 be pushed into the stack in the order they appear from left to right.

Assume S and X indicate the push and pop operations respectively. The operation sequence SSSSSSXXXXXX outputs (pops) the integers in the order: 6, 5, 4, 3, 2, 1.

State an operation sequence that outputs the integers in the following order: 3, 2, 5, 6, 4, 1.

(6 marks)

(b) A queue may be implemented using an array or a linked list. Discuss one advantage and one disadvantage of each implementation.

(8 marks)

Note: Question No. 4 continues on Page 5

(c) Consider a binary tree BT that has 86 nodes. Answer the following questions about BT and explain the reasoning behind all your answers.

(i) What is the minimum height of BT?

(2 marks)

(ii) What is the minimum number of leaf nodes in BT?

(3 marks)

(iii) Suppose BT is also a binary search tree, and has a height of 85. What does that tell us about the items?

(3 marks)

(iv) Explain whether a binary search tree as described in 4(c)(iii) allows for efficient retrieval of the items stored in it.

(3 marks)

END OF PAPER

NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 2 EXAMINATION 2013-2014

CE1007/CZ1007 – DATA STRUCTURES

Apr/May 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) What is the output of the following code?

```
#include <stdio.h>
void fun(int n, int *r) ;
int main()
{
   int m=10,r=1;
   fun(m,&r);
   printf("m=%d,r=%d\n",m,r);
   return 0;
}
void fun(int n, int *r)
{
   *r*=n;
   if (n>0)
      fun(n-4,r);
   printf("n=%d,*r=%d\n",n,*r);
}
```

(7 marks)

Note: Question No. 1 continues on Page 2

(b) In the following program, the function `readin()` reads a number of people's names and their corresponding telephone numbers, passes the data to the calling function via the parameter `p`, and returns the number of names that have been entered. The character '#' is used to indicate the end of user input. The function `search()` finds the telephone number of an input name `target`, and then prints the name and telephone number on the screen. If the input name cannot be found, then it will print an appropriate error message. Write the two missing code segments `(i)` and `(ii)` in the program. Note that you should not declare any other variables in the program.

```
#include <stdio.h>
#include <string.h>
#define MAX 100
typedef struct {
   char name[10];
   char telno[10];
} PhoneBk;
int readin(PhoneBk *p);
void search(PhoneBk *p, int size, char *target);
int main()
{
   PhoneBk s[MAX];
   char t[10];
   int size;
   size = readin(s);
   printf("Enter search name: ");
   gets(t);
   search(s,size,t);
   return 0;
}
int readin(PhoneBk *p)
{
   int size=0;
   /* (i) missing code segment here */
   return size;
}
void search(PhoneBk *p, int size, char *target)
{
   int i;
   /* (ii) missing code segment here */
}
```

(10 marks)

Note: Question No. 1 continues on Page 3

(c) The following program reads a character string and determines whether or not it is a *palindrome*. A palindrome is a sequence of characters that reads the same forwards and backwards. For example, *abba* and *abcba* are palindromes, but *abcd* is not. Write the missing code segment in the program. Note that you should not declare any other variables in the program.

```
#include <stdio.h>
#include <string.h>
int main()
{
   char str[80],*p1,*p2;
   int n;
   gets(str);
   n=strlen(str);
   /* missing code segment here */
   return 0;
}
```

(8 marks)

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

```
#include <stdio.h>
int main()
{
   int a[]={2,4,6,8,10},i;
   int *ptr[]={&a[0],&a[1],&a[2],&a[3],&a[4]};
   int *pa,**pb;
   pa=a;
   printf("%d,",*++pa);
   printf("%d\n",*pa++);
   for (i=0;i<5;i++) {
      a[i]=a[i]/2+a[i];
      printf("%d,",a[i]);
   }
   pb=ptr;
   printf("\n%d,", *(*(pb+2)));
   printf("%d", *(*(++pb)));
   return 0;
}
```

(8 marks)

Note: Question No. 2 continues on Page 4

(b) In the following program, the function `max()` takes a one-dimensional array `m` of integers as a parameter. The function returns the largest and second largest numbers of the array to the `main()` function through the two parameters `max1` and `max2` respectively. For example, if the array `a[]={1,7,8,6,9,4,5,2,3}`, then `max1` is 9, and `max2` is 8. Write the missing code segment in the program. Note that you should not declare any other variables in the program.

```
#include <stdio.h>
#define SIZE 9
void max(int m[SIZE], int *max1, int *max2);
int main()
{
    int max1,max2;
    int a[]={1,7,8,6,9,4,5,2,3};
    max(a,&max1,&max2);
    printf("max1=%d,max2=%d\n",max1,max2);
    return 0;
}
void max(int m[SIZE], int *max1, int *max2)
{
    int i;
    *max1=m[0]>m[1]?m[0]:m[1];
    *max2=m[0]>m[1]?m[1]:m[0];
    /* missing code segment here */
}
```

(7 marks)

(c) The following program computes the sum of the elements of the two diagonals in a 3x3 two-dimensional array of integers. For example, if the array `a[3][3]={1,2,3,4,5,6,7,8,9}`, then `sum1` is 1+5+9=15, and `sum2` is 3+5+7=15. Write the missing code segment in the program. Note that you should not declare any other variables in the program.

```
#include <stdio.h>
int main()
{
    int a[3][3]={1,2,3,4,5,6,7,8,9};
    int i,j,sum1=0,sum2=0;
    /* missing code segment here */
    printf("sum1=%d,sum2=%d\n",sum1,sum2);
}
```

(10 marks)

3. (a) Stacks and queues are special types of list data structures. Describe how stacks and queues differ from regular linked list data structures.

(3 marks)

(b) Describe the difference between stacks and queues.

(3 marks)

(c) Consider a singly-linked list that was built using doubly-linked list nodes. Write a C function `repairDblLinkedList()` that traverses such a linked list and repairs any broken links between nodes as depicted in Figure Q3. Pointers corresponding to broken links have been set to `NULL`. The function should return the number of broken links that were repaired. The function prototype is given as follows:

```
int repairDblLinkedList(DblLinkedList *ll);
```

(i) Before repair (ii) After repair

Figure Q3

(12 marks)

(d) Consider the following representation of a polynomial of degree n, where a_i, x and n are all integer values:

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_1 x + a_0$$

Explain, using any necessary diagrams and/or C code, how you would use a linked list data structure to store a given polynomial.

(7 marks)

4. (a) Describe how binary search trees differ from general binary tree data structures.

(4 marks)

Note: Question No. 4 continues on Page 6

(b) Consider the binary tree T shown in Figure Q4. Use pre-order, in-order and post-order methods to traverse T. For each traversal order, do the following:

(i) State the print order of the values stored in each node.

(6 marks)

(ii) Show by drawing arrows on a copy of the diagram, the order in which nodes are visited during the tree traversal progress. Only draw an arrow when an actual node is visited.

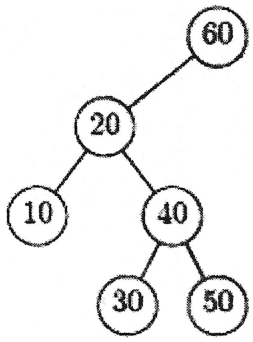


Figure Q4

(6 marks)

(c) Write a C function `printOddValDesc()` that accepts a pointer to the root node of a binary search tree B and prints any odd numbers stored in B in decreasing order of magnitude. The function prototype is given as follows:

```
int printOddValDesc(btnode *node);
```

(9 marks)

END OF PAPER

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`.

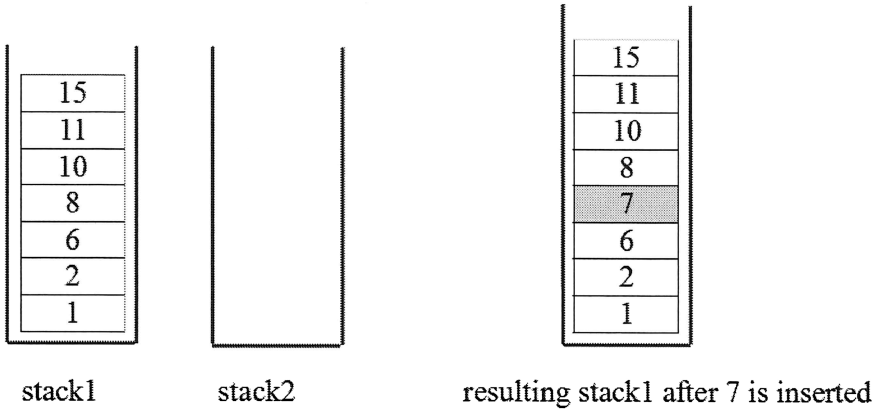


Figure Q3

(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)

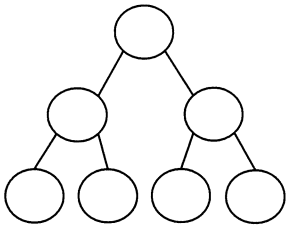
4. (a) Draw a binary search tree that may be generated from the integer set {1,2,3,4,5,6,7}. Describe the concept of a *binary search tree*.

(5 marks)

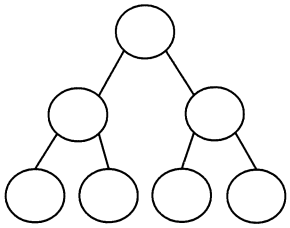
(b) Three different binary trees with missing node data are shown in Figure Q4. Each node in the tree stores a character, which belongs to the set C = {'*a*', '*b*', '*c*', '*d*', '*e*', '*f*', '*g*'}. Draw these three trees and choose a proper character selected from the set C for every node, so that 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)

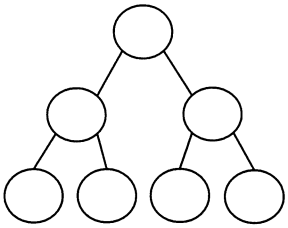
Note: Question No. 4 continues on Page 6



(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

NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 2 EXAMINATION 2014-2015

CE1007/CZ1007 – DATA STRUCTURES

Apr/May 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() {
   int a[4][4]={  {1,2,-3,-4},
                  {5,-4,-3,2},
                  {2,0,-3,2},
                  {-6,5,-4,3}  };
   int i,j,s=0;
   for (i=0; i<4; i++) {
      for (j=0; j<4; j++) {
         if (a[i][j]<0) continue;
         if (a[i][j]==0) break;
         s+=a[i][j];
      }
      printf("%d\n",s);
   }
   return 0;
}
```

(4 marks)

Note: Question No. 1 continues on Page 2

(ii)
```
#include <stdio.h>
void f1(int *a, int *b);
int main() {
   int a=1,b=2,c=3;
   f1(&a,&b); printf("%d,%d\n",a,b);
   f1(&b,&c); printf("%d,%d\n",b,c);
   f1(&c,&a); printf("%d,%d\n",c,a);
   return 0;
}
void f1(int *a, int *b) {
   *a=*a+*b;
   *b=*a-*b;
   *a=*a-*b;
}
```

(4 marks)

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

(i)
```
/* This program reads in a sequence of
   positive numbers, terminated by 0, and
   computes and prints the average of the
   positive numbers (excluding 0). */
 1 #include <stdio.h>
 2 int main() {
 3    int s[100];
 4    int i=0,c=0;
 5    int sum=0;
 6    do {
 7       scanf("%d",&s[i]);
 8    } while (s[i++] != 0);
 9    i=0;
10    while (s[i] = 0) {
11       printf("%d ",s[i]);
12       if (s[i]>0) {
13          sum += s[i];
14          c++;
15       }
16       i++;
17    }
18    sum \= c;
19    printf("%f\n",c);
20    return 0;
21 }
```

(4 marks)

Note: Question No. 1 continues on Page 3

(ii)

```
/* This program finds the smallest string
   from an array of 5 strings according to
   the ascending alphabetical order. */
#include <stdio.h>
#include <string.h>
int main() {
   char *name[5] = {"Windows", "Word",
      "Excel", "Foxpro", "Visual Basic"};
   char temp;
   int i;
   temp=name[0];
   for (i=1; i<=5; i++)
      if (temp>name[i])
         temp=name[i];
   printf("%s\n",*temp);
   return 0;
}
```

(4 marks)

(c) The following program reads in two character strings `a` and `b`, concatenates them to form a new string, and prints the resultant string to the screen. For example, if `a` = "Nanyang " and `b` = "Technological University", then the new string is "Nanyang Technological University". Write the missing code segment in the program. Note that you should not use any standard C string functions and declare any other variables in the program.

```
#include <stdio.h>
int main() {
   char a[80],b[80],*p,*q;

   gets(a);
   gets(b);
   /* missing code segment here */
   return 0;
}
```

(9 marks)

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

(i)
```
#include <stdio.h>
int f2(int x, int *y);
int main() {
   int x=3, y=5;
   printf("%d\n", f2(x,&y));
   return 0;
}
int f2(int x, int *y)
{
   if (x<=0 || *y<=0)
      return 1;
   else {
      *y -=2;
      return 5*f2(x-1,y);
   }
}
```

(3 marks)

(ii)
```
#include <stdio.h>
long f3(int n);
int main(){
   printf("f3(): %d\n", f3(5));
   return 0;
}
long f3(int n){
   switch (n) {
      case 0:
         return 0;
      case 1: case 2:
         return 1;
   }
   return (f3(n-1)+f3(n-2));
}
```

(4 marks)

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

Note: Question No. 2 continues on Page 5

(i)

```
/* f4() counts the number of upper and
   lower case letters from a string. */
#include <stdio.h>
void f4(char *s, int a, int b);
int main() {
   char s[80]; int upper=0,lower=0;
   gets(s); f4(s,&upper,&lower);
   printf("%d %d\n",upper,lower);
   return 0;
}
void f4(char *s, int a, int b) {
   while (*s) {
      if (*s>='A' && *s<='Z') a++;
      if (*s>='a' && *s<='z') b++;
      s++;
   }
}
```

(4 marks)

(ii)

```
/* f5() finds the largest character from
   a string, and moves it to the beginning
   of the string. E.g., if str is "aedcb"
   before calling f5(), then str will be
   "eadcb" after executing f5(). */
#include <stdio.h>
void f5(char *p);
int main() {
   char *str;
   gets(str); f5(str); puts(str);
   return 0;
}
void f5(char *p) {
   char max,*q; int i=0;
   while (p[i] != '\0') {
      if (max<p[i]) {
         max=p[i];
         p=q+i;
      }
      i++;
   }
   while (q<p) {
      q=*(q-1);
      q--;
   }
   p[0]=max;
}
```

(6 marks)

Note: Question No. 2 continues on Page 6

(c) In the following program, the **recursive** function `rLookup()` returns the subscript of the first appearance of a target number in the array. The function takes in three arguments `ar[]`, `n` and `target`, which are an array of integers, an integer specifying the size of the array and the target number respectively. For example, if the array contains, in order, 1, 2, 3, 4, 5, and the target number is 2, then `rLookup()` will return 1. If the target number is not in the array, then `rLookup()` will return –1. Write the missing code segment in the program. Note that you should not declare any other variables in the program.

```
#include <stdio.h>
int rLookup(int ar[], int n, int target);
int main(){
   int a[80];
   int target, i, size;
   printf("Enter array size: ");
   scanf("%d", &size);
   printf("Enter %d numbers: ", size);
   for (i=0; i < size; i++)
      scanf("%d", &a[i]);
   printf("Enter the target number: ");
   scanf("%d", &target);
   printf("%d", rLookup(a, size, target));
   return 0;
}
int rLookup(int ar[], int n, int target){
   int i;
   if (n == 1) {
      if (ar[0] == target)
         return 0;
      else
         return -1;
   }
   else {
      /* missing code segment here */
   }
}
```

(8 marks)

3. (a) The following program asks the user how many integers will be entered, and then reads in the integers from the user.

(i) Fill the missing codes of Line 8 and Line 13.

(4 marks)

(ii) If Line 11 is replaced by "`for (i=0; i<=n; i++)`", what problem will it cause?

(2 marks)

```
1   #include <stdlib.h>
2   #include <stdio.h>
3   int main(){
4      int n, i;
5      int *intArr;
6      printf("How many integers do you have? ");
7      scanf("%d", &n);
8      intArr = malloc(_____________________________);
9      if (intArr == NULL) printf("error.\n");
10    /*Loop over array & store integers entered*/
11     for (i=0; i<n; i++)
12         scanf("%d",&intArr[i]);
13     _________________;/*free the memory space of
                          intArr*/
14     return 0;
15  }
```

(b) Consider an empty stack S of integers. Let integers be pushed into the stack S in the order of **1→2→3**. Assume U and O indicate the push() and pop() operations respectively. For example, the operation sequence UOUUOO for the stack S outputs (pops) integers in the order: 1, 3, 2.

(i) What is the output if the operation sequence is UUOO?

(2 marks)

(ii) List all possible *6-operation sequences* and the corresponding outputs. A *6-operation sequence* means a sequence with 6 operations, e.g., UUUOOO.

(5 marks)

(iii) State the main difference between a queue and a stack.

(2 marks)

Note: Question No. 3 continues on Page 8

(c) Write the missing code segment of the function `swap2n()` that accepts the head pointer of a linked list of integers. It swaps the **2nd node** with the **last node** of the linked list and **returns the number of nodes** in the linked list. **Swapping of data is not allowed, only pointers should be changed.** Suppose the linked list has at least **4 nodes**. You are **NOT** allowed to call any linked list functions.

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

int swap2n(ListNode *head){
   int size=0;          /* size is the number of
                           nodes in the linked list*/
   ListNode *p=head;  /* pointer p is used for
                      traversal of the linked list*/
   ListNode *tmp;     /*for temporary store of some
                      pointers when swapping nodes*/

   /* missing code segment here */

   return size;
}
```

(10 marks)

4. (a) Two trees are shown in Figure Q4(1) and Figure Q4(2).

(i) Is the tree in Figure Q4(1) a binary search tree? Justify your answer.

(3 marks)

(ii) Is the tree in Figure Q4(2) a binary search tree? Justify your answer.

(3 marks)

Note: Question No. 4 continues on Page 9

(1) (2)

Figure Q4

(b) A binary tree has 4 nodes that store 'A', 'B', 'C' and 'D' respectively. The in-order traversal visits the tree in the order of ABCD, and the pre-order traversal visits the tree in the order of CABD.

(i) Draw the binary tree structure.

(5 marks)

(ii) Based on the binary tree you draw, what is the node visit order by the post-order traversal?

(3 marks)

(c) Write a **recursive** C function `printParent()` that accepts a pointer `node` to the **root** node and a pointer `x` to a **non-root** node of a binary search tree. It finds the parent node of `x` and prints the parent node's item value. It returns **1** when it is able to find the parent node of `x`, otherwise it returns **0**. The function prototype is given as follows:

```
int printParent(BTNode *node, BTNode *x);
```

A `BTNode` structure is defined as follows:

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

(11 marks)

END OF PAPER

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.

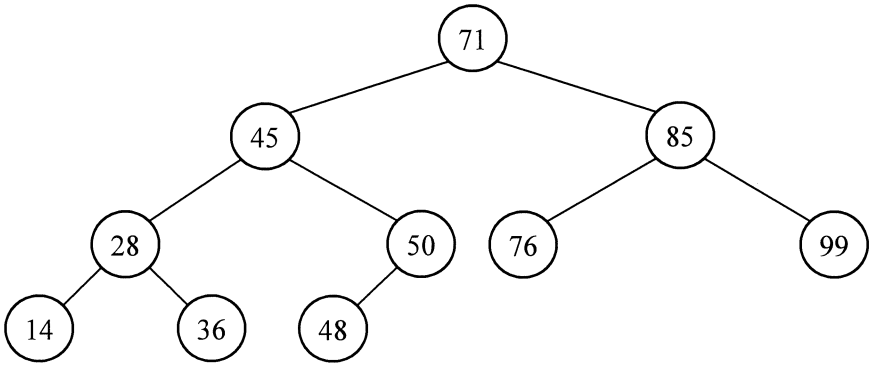


Figure Q4

(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

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