Name: . . . . . .

ID Number: . . . . . .

COMP206: Program and Data Structures
Mid-Term Test

10 August 2005

Instructions

- Maximum time: **60 minutes**.
- Answer **all** the questions.
- There are 60 marks in total.
- Write your answers in the test paper and hand in all sheets.
- Paper translation dictionaries are allowed.
- Non-programmable calculators are allowed.
- Every box with a heavy outline requires an answer.
- Page 15 shows some standard functions for data file processing in C.

Questions	**Marks**	
1. C and C++ Basics	[20]	
2. C Pointers, Arrays, and Strings	[10]	
3. C Function Calls	[10]	
4. C Data Files	[10]	
5. Dynamic Data Structures in C	[10]	
Total Marks	[60]	

Question 1. C and C++ Basics [20 marks]

(a) [4 marks] Evaluate the following C code expressions and give the result of each expression after the symbol "$\Longrightarrow$".

(i) `6 % 2 - 9 / 2` $\Longrightarrow$

(ii) `3 * 5 >> 1` $\Longrightarrow$

(iii) `20 != 5 << 2` $\Longrightarrow$

(iv) `7 % 2 ? 6 * 4 : 11 / 2` $\Longrightarrow$

(b) [6 marks] Describe the output that will be generated by each of the following C programs in the given boxes.

(i)

```
#include <stdio.h>

int main()
{
  int i = 0, x = 0;

  do {
    if (i % 5 == 0) {
      x++;
      printf("%d ", x);
    }
    i++;
  } while (i < 20);
  printf("\nx = %d\n", x);
  return 0;
}
```

(ii)

```
#include <stdio.h>

int main()
{
  int i = 0, x = 0;

  for (i = 1; i < 5; i++) {
    if (i % 2 == 1)
      x += i;
    else
      x--;
    printf("%d ", x);
    continue;
  }
  printf("\nx = %d\n", x);
  return 0;
}
```

(c) [5 marks] Write the outputs of the following C program (`static.c`) in the given box.

```
/* static.c */
#include <stdio.h>

int mystery(int);

int main()
{
  int x = 0;
  int i = 0;

  for (i = 0; i < 4; i++) {
    x += mystery(i);
    printf("i = %5d\tx = %5d\n", i, x);
  }

  return 0;
}

int mystery(int k)
{
  static int m;  /* Pay attention here!!! */

  return ((k) ? (m += 2) : (m = 0));
}
```

(d) [5 marks] Consider the following C++ program segments `time2.h` and `time2.cpp`. Identify five distinct errors in the `time2.h` and `time2.cpp` files and make your corrections in the code segments below.

```
// time2.h
#ifndef TIME2.H
#define TIME2.H

class Time {
public:
  Time();   // Time constructor
  void setTime (int, int, int); // Set hour, minute, second
  void printMilitary(); // print militaty time format
  void printStandard (); // print standard time format
private:
  int hour; // 0-23
  int minute; // -59
  int second; //0-59
}

#endif
//-----------------------------------------------


// time2.cpp
#include <iostream>
#include "time2.h"

void Time::Time() {hour = minute = second =0; }

void Time::setTime(int h, int m, int s)
{
  hour = (h >= 0 && h < 24)? h : 0;
  minute = (m >= 0 && m < 60) ? m : 0;
  second = (s >= 0 && s < 60) ? s : 0;
}

void printMilitary()
{
  cout << (hour < 10 ? "0": "") << hour << ":"
       << (minute < 10 ? "0": "") << minute;
}

void Time::printStandard()
{
  cout << ((hour == 0 || hour == 12) ? 12 : hour % 12)
       << ":" << (minute < 10 ? "0": "") << minute
       << ":" << (second < 10 ? "0": "") << second
       << (hour < 12 ? " AM" : " PM");
}
```

Question 2. C Pointer, Array, String [10 marks]

Given the C variable declarations below:

```
int c = 4;
int *p;
char a[] = "Hello World";
float matrix[2][3] = {{1.0, 2.0, 3.0}, {4.0, 5.0, 6.0}};
```

Evaluate the following C expressions and give the results of each expression after the symbol "$\Longrightarrow$".

(a) `a` $\Longrightarrow$

(b) `*(a + 4)` $\Longrightarrow$

(c) `*a + 3` $\Longrightarrow$

(d) `*(p = &c)` $\Longrightarrow$

(e) `matrix[1][2]` $\Longrightarrow$

(f) `*matrix[1] + 2` $\Longrightarrow$

(g) `(*matrix + 1)[3]` $\Longrightarrow$

(h) `(*matrix)[1]` $\Longrightarrow$

(i) `(*(matrix + 1))[0]` $\Longrightarrow$

(j) `*(*matrix + 2)` $\Longrightarrow$

Question 3. C Function Calls [10 marks]

Write two C functions (`fact1` and `fact2`) to handle the factorial problem:

$$fact(n) = 1 \times 2 \times 3 \times ... \times n$$

The main function and the prototypes of the two functions to be written by you are given below. Assume that there is no overflow here.

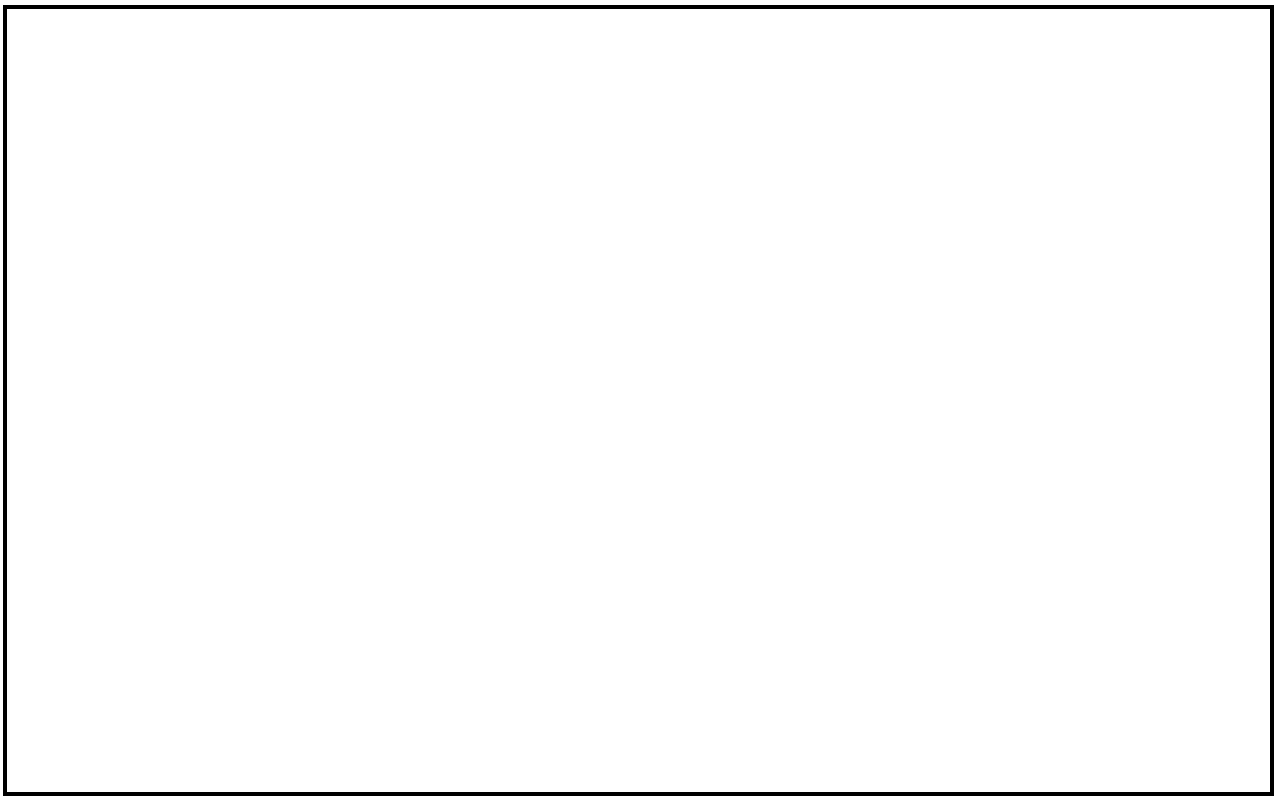
```
#include <stdio.h>

double fact1(int);
void fact2(int, double *);

int main()
{
  int m;
  double x = 1.0;

  printf("Enter an integer: ");
  scanf("%d", &m);

  printf("The factorial of %4d computed by fact1 is %10.1f\n", m, fact1(m));

  fact2(m, &x);
  printf("The factorial of %4d computed by fact2 is %10.1f\n", m, x);

  return 0;
}
```

Question 4. C Data Files [10 marks]

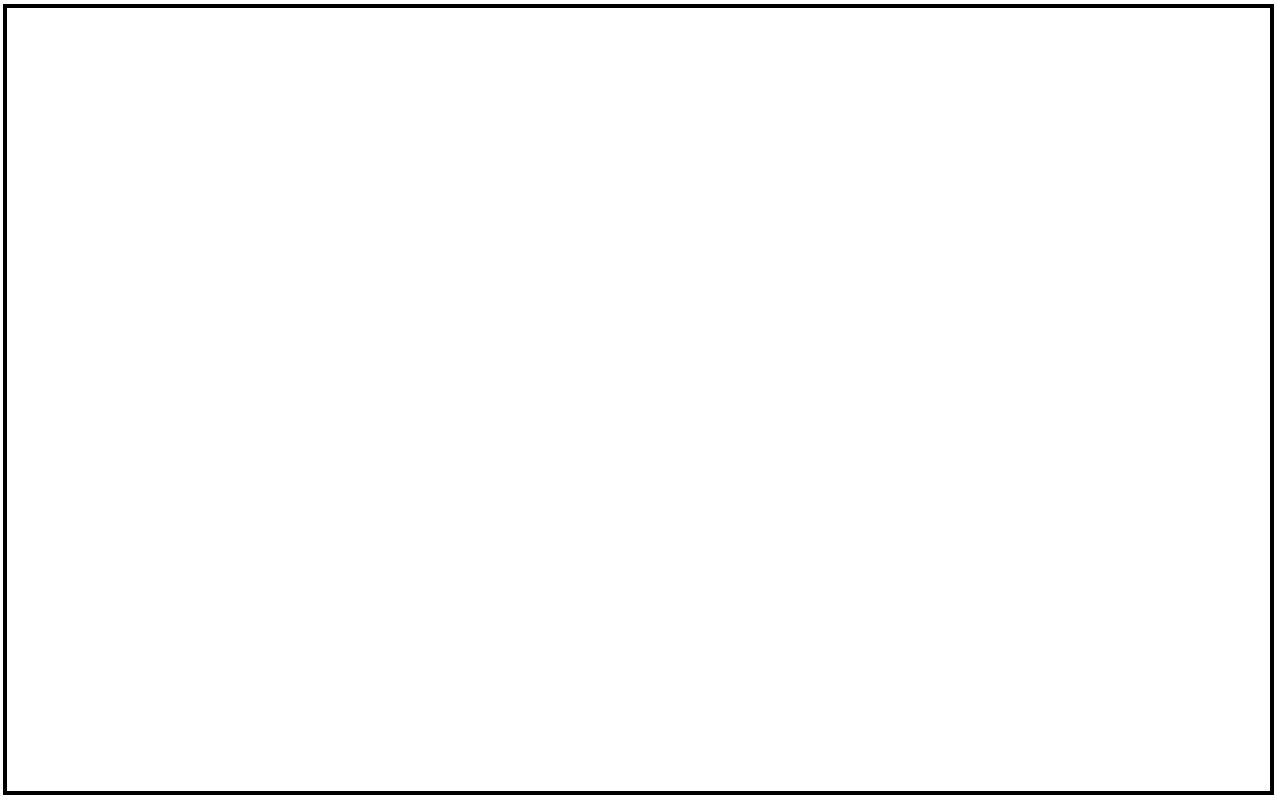
The following C program takes five records of type `Person` from the keyboard, uses `fopen, fscanf, fprintf`, and `fclose` to create a formatted file called "aaa.dat" by writing all the records into the file then the program reads all the five records from the file.

```
#include <stdio.h>
#include <ctype.h>

typedef struct person {
  char name[10];
  int age;
} Person;

int main()
{
  FILE *fp1, *fp2;
  Person p1[5], p2[5];
  int i;

  fp1 = fopen("aaa.dat", "w");
  for (i = 0; i < 5; i++) {
    scanf(" %s %d", p1[i].name, &p1[i].age);
    fprintf(fp1, "\n%s \t %7d", p1[i].name, p1[i].age);
  } /* for i */
  fclose(fp1);

  fp2 = fopen("aaa.dat", "r");
  for (i = 0; i < 5; i++)
    fscanf(fp2, " %s %d", p2[i].name, &p2[i].age);
  fclose(fp2);

  return 0;
}
```

(a) [5 marks] Rewrite the program using `fopen, fread, fwrite` and `fclose` to implement the corresponding steps to process a binary file called "bbb.dat". You should write your `main` function only; omitting the structure definition and header files.



(b) [5 marks] Rewrite the program using low level file operations `open, read, write` and `close` to implement the corresponding steps to process a binary file called "ccc.dat". You can write your `main` function only and omit the structure definition and header files.



Question 5. Dynamic Data Structures in C [10 marks]

Given the following linked list definition, function prototypes and the `main` function we have discussed during lectures:

```
typedef struct person {
  char name[20];
  struct person *next;
} Person;

typedef Person * PersonPtr;
PersonPtr newPersonNode(); /* memory allocation  for a node */
void printlist(PersonPtr); /* print/traverse a linked list */
int count(PersonPtr);     /* count the number of elements in the list */
PersonPtr findElement(PersonPtr, char *);  /* Search for a given
                         element in the list */
void insertElement(PersonPtr, char *); /* insert an element to
                         the list after a given node */
void deleteElement(PersonPtr);  /* delete an element from the
                         list after a given node */
int main()
{
  PersonPtr Personlist;
  PersonPtr current, previous;
  char in_name[20];

  /* Create a linked list: aaa->bbb->ccc->ddd->eee */
  Personlist = NULL;
  while (strlen(gets(in_name))!= 0) {
    if ((current = newPersonNode()) == NULL) exit(1);
    strcpy(current->name, in_name);
    if (Personlist == NULL) Personlist = current;
    else previous->next = current;
    previous = current;
  }
  previous->next = NULL;

  /* print a linked list */
  printlist(Personlist);

  /* Count the number of elements in the list */
  printf("\nThere are %d elements in the list\n",count(Personlist));

  /* Find an element in the list */
  current = findElement(Personlist, "bbb");

  /* insert a person "James" after the Person "bbb" */
  insertElement(current, "James");
  printlist(Personlist);

  /* delete a person after the person "ccc" */
  current = findElement(Personlist, "ccc");
  deleteElement(current);
  printlist(Personlist);

  return 0;
}
```

(a) [3 marks] Write C code to implement function `newPersonNode` so that the storage/memory of a node can be allocated.

(b) [3 marks] Write C code to implement function `printlist` so that all elements stored in the list can be printed out.

(c) [4 marks] Write C code to implement function `insertElement` so that a new person's record can be added into the list after a given node.

A C Standard Functions in Data File Processing

```
FILE *fopen(const char * restrict path, const char * restrict mode);
int   fclose(FILE *stream);
size_t fread(void * restrict ptr, size_t size, size_t nmemb,
         FILE * restrict stream);
size_t fwrite(const void * restrict ptr, size_t size, size_t nmemb,
         FILE * restrict stream);
int  open(const char *path, int flags, mode_t mode);
int  close(int d);
ssize_t read(int d, void *buf, size_t nbytes);
ssize_t write(int d, const void *buf, size_t nbytes);
```

****