Name (Pinyin): _____ **姓名: _____**

S&T2024 Advanced C Programming Language Mock Test (Third Checkpoint)

Time allowed: 1 hour and 30 minutes

INSTRUCTIONS

1. This is a **CLOSED BOOK** examination.
2. This examination paper contains **6 questions** and 8 pages. Answer **all** questions.
3. The full score for each question is as follows:
   Question 1: 5 Question 2: 5 Question 3: 5 Question 4: 5
   Question 5: 30 Question 6: 50
4. Answers to the questions are to be written in a seperate booklet.
5. Students are **NOT** allowed to bring any programmable calculator, machine translator, or dictionary with them. If you have any of these items, you are to surrender them before the test starts. **Violation of this regulation will be severely dealt with and may result in an expulsion from the bridging course**.
6. Students are also **NOT** allowed to bring any paper to the Examination Hall.
7. All the working steps must be shown.
8. This test paper will be collected at the end of the examination.

No.	Score 1 (Marker)	Score 2 (Checker)
1		
2		
3		
4		
5		
6		

1. The following program is named as **test.c**.

```
#include<stdio.h>

main(int argc,char *argv[])
{
  printf ("\n (i) %d", argc);
  printf ("  (ii) %d", argv[1]);
  printf ("  (iii) %c", *argv[3]+2);
  printf ("  (iv) %s", argv[3]+2);

  return 0;
}
```

It is run at the command prompt as follows:

D:\>**test one two three four**

Write down the screen output. **(5 Marks)**

2. If DEV C is used to compile the following program, the range of 32-bit signed integer is from -2,147,483,648 (inclusive) to 2,147,483,647 (inclusive). Write down the screen output of the following program.

```
#include<stdio.h>

main()
{
  int size;
  struct
  {
    char name[5];
    int Part1;
    int Part2;
  } demo;

  size=sizeof(demo);
  printf("Size of struct is %d bytes.  \n",size);
  demo.Part1 = -1;
  demo.Part2 = -3;

  printf("Part1=%u, Part2=%u\n",demo.Part1,demo.Part2);

  return 0;
}
```

(5 Marks)

3. Re-write the following C programming **without** using the ==, >, >=, <, <= and ! operators to compare two numbers.

```
void main ()
{
  int a, b;

  printf ("Enter two integers: ");
  scanf ("%d%d", &a, &b);

  if (a==b)
    printf ("Equal");
  else
    printf ("Not Equal");
}
```

(5 Marks)

4. Write down the output of the following C program.

```
#include<stdio.h>

int main()
{
  char *p, *q, r;

  p=q=&r;
  *p = 65;
  (*p)++;

  printf(" %c", *q);

  return 0;
}
```

(5 Marks)

5. Bit-wise transformation technique can be used to scramble data for security purpose. Assume the following declarations:

```
unsigned this1, result;
```

Let the bits of the unsigned integer **this1** be labeled as $\mathbf{b_{31}b_{30}}$ ...$\mathbf{b_2b_1b_0}$, and let its contents be $(\mathbf{h_7h_6h_5h_4h_3h_2h_1h_0})_{(16)}$, i.e., $\mathbf{h_7h_6h_5h_4h_3h_2h_1h_0}$ is a hexadecimal number, such as $345abd7f_{(16)}$, $ffffffff_{(16)}$, etc.

Our encryption goes as follows. Given a hexadecimal number $\mathbf{h_7h_6h_5h_4h_3h_2h_1h_0}$, the encryption will produce the **result** with $\mathbf{h_4h_5h_3h_2h_1h_0h_6h_7}$.

(i) Complete the encode function that accepts an unsigned integer as an input argument, and performs the above transformation on the unsigned integer by bit manipulation operators. The result is then returned to the caller. The function header is given as follows:

```
unsigned encode (unsigned this1)
```

For example, if this1 = fedcba98$_{(16)}$, the encode function should return cdba98ef$_{(16)}$.

(ii) Complete the decode function to perform the inverse transformation to recover the original unsigned integer by bit manipulation operators. The input parameter is the encoded unsigned integer. The original unsigned integer should be returned to the caller. The function header is given as follows:

```
unsigned decode (unsigned result)
```

For example, if result = 45321067$_{(16)}$, the decode function should return 76543210$_{(16)}$.

(30 Marks)

6. In one of the lab exercise in phase-1, we worked on the text file stated in the lab handout as follows:

A text file named as wage.inf (unzip from lab5.zip) contains the information of all employees in a small factory. The file has the following contents:

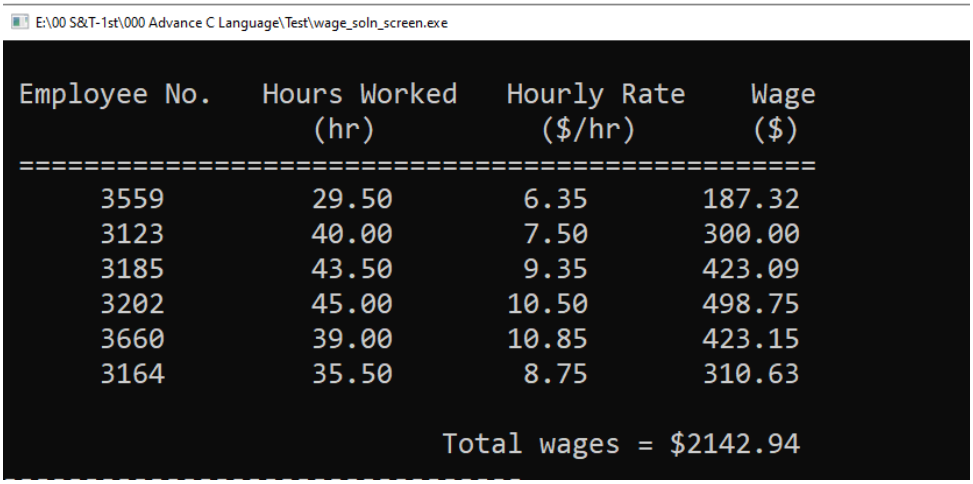
i) Employee name of not more than 13 characters (column 1 to 13)
ii) Employee number (column 15 to 18)
iii) Gender (column 20)
iv) Hourly rate (floating-point number)
v) Number of hours worked (floating-point number)

The input file has the **new contents** and the employee numbers are **not sorted**.

```
Henry Loh      3559 M   6.35  29.5
Tan Ai Lee     3123 F   7.50  40
Chan Meng      3185 M   9.35  43.5
Tan Cheng Lee  3202 M   10.50 45
Lim Gao        3660 M   10.85 39
Wong Alice     3164 F   8.75  35.5
```

Input File (new contents) : wage.inf

We have also written a C program named as **wage.c** to compute the wages of each employee and to sum up all the wages. The regular pay is computed by Hourly rate * Number of hours worked for up to 40 hours. For those hours greater than 40, a multiplier of 1.5 is used to compute the overtime pay. The wage for each employee is the sum of regular pay and overtime pay. For example, the wage received by `Chan Meng` is computed as 9.35*40 + 9.35*(43.5-40)*1.5 = 423.09. In this version we print the output on the screen as follows:



Screen Output (Employee numbers are not sorted)

The complete program for the above screen output (employee numbers are not sorted) is given as follows:

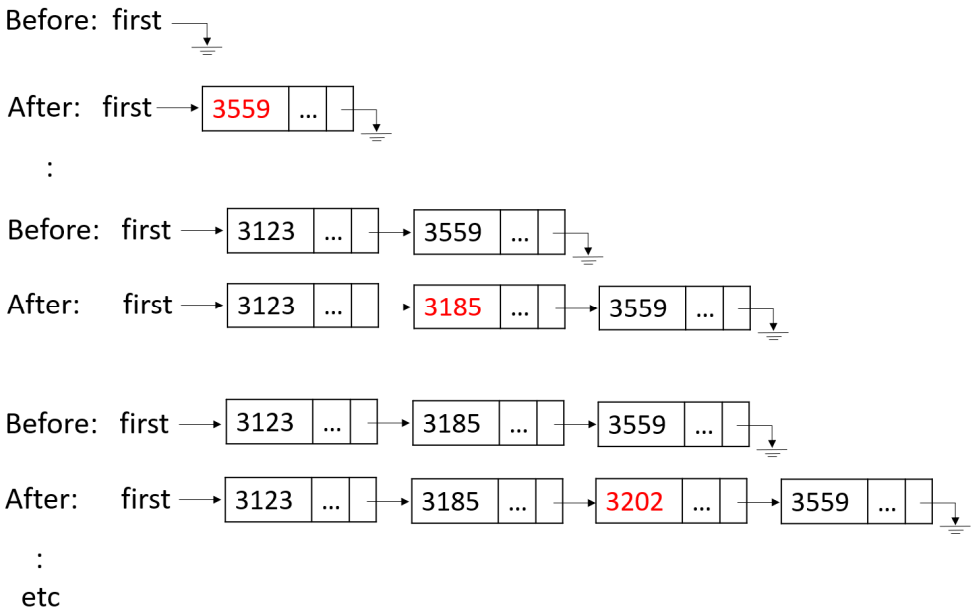
```
# include <stdio.h>

main(void)
{
 char name [14];
 int emp_no;
 char gender;
 float rate, hr, pay, total=0;
 FILE *indata;

 indata  = fopen ("wage.inf","r");

 if (indata==NULL)
 {
   printf ("\n wage.inf cannot be found."); exit (1);
 }

 printf ("\n Employee No.   Hours Worked   Hourly Rate    Wage");
 printf ("\n                (hr)        ($/hr)      ($)");
 printf ("\n ================================================");

 // 13 = 14 - 1
 while ( (fgets (name,14,indata) !=NULL) &&
         (fscanf (indata,"%d %c%f%f%*c",&emp_no,&gender,&rate,&hr)==4) )
 {
     printf ("\n     %4d       %6.2f     %6.2f   ",    emp_no, hr,  rate );

     if (hr <= 40) pay = hr*rate;
     else  pay = rate *40 + rate*(hr-40)*1.5;
```

```
    printf ("    %6.2f ", pay);

    total +=pay;
  }

  printf ("\n\n                    Total wages = $%-8.2f", total);
  fclose (indata);

  return 0;
}
```

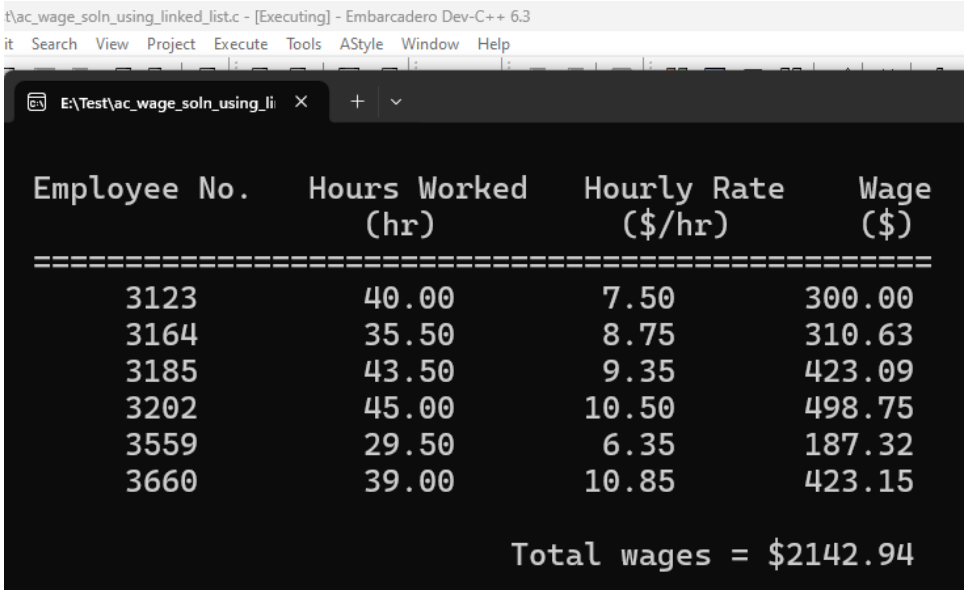
In this test we want to print the wages in **ascending order of the employee number** but we do not want to make use of any sorting algorithm or to revert to the old contents (given in phase 1) in wage.inf. Instead we will make use of the linked list to create the wage record and insert the node to the correct place one by one in the ascending order of the employee number. The node is defined as follows:

```
typedef struct employee
{
   char name [14];
   int emp_no;
   char gender;
   float rate, hr;
   struct employee *next;
} node;
```

When creating the linked list in your solution, you have to take care of the sequence of the employee number to be inserted. For examples:



Finally, when the nodes are printed from the first pointer, the screen output should be as follows:



Screen Output produced by the final link list
(in sequence of increasing employee number)

Complete the following programme. You can declare additional identifiers if needed.

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

int main(void)
{

  typedef struct employee
  {
    char name [14];
    int emp_no;
    char gender;
    float rate, hr;
    struct employee *next;
  } node;

  float pay, total=0;
  FILE *indata;

  node this1;
  node *temp, *first=NULL;

  indata  = fopen ("wage.inf","r");

  if (indata==NULL)
  {
    printf ("\n wage.inf cannot be found.");
    exit (1);
```

```
  }

  printf ("\n Employee No.   Hours Worked   Hourly Rate    Wage");
  printf ("\n                   (hr)          ($/hr)       ($)");
  printf ("\n ================================================");

  while ( (fgets (this1.name,14,indata) !=NULL) &&
    (fscanf (indata,"%d %c%f%f%*c",&(this1.emp_no), &(this1.gender),
                                    &(this1.rate),&(this1.hr) )==4) )
  {
      // Fill in this portion only.
      // You can declare additional identifiers if needed
  }

  // Finally, print out the whole linked list
  temp = first;
  while (temp!=NULL)
  {
     printf ("\n      %4d        %6.2f      %6.2f   ",
                                temp->emp_no, temp->hr, temp->rate );
     if (temp->hr <= 40)
       pay = temp->hr * temp->rate;
     else
       pay = temp->rate *40 + temp->rate*(temp->hr-40)*1.5;

     printf ("    %6.2f ", pay);

     total +=pay;
     temp = temp->next;
  }

  printf ("\n\n                            Total wages = $%-8.2f", total);
  fclose (indata);
  return 0;
}
```

(50 Marks)