INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Programming and Data Structures (CS10003) **Spring Semester 2021-22**
Long Test-2 (Part B) **Marks=50**

21 June 2022 from 10:20 to 11:30

Instructions:

1. *Write your answers on paper. Answers must be handwritten. Typed or written answers using an electronic device are not allowed.*
2. *Write your name and roll number on each page. Write page numbers for each page.*
3. *Answer all five questions. All parts of a question must be answered in the same place.*
4. *Scan all pages and collate. Create a single PDF file (of size <= 10 MB). You could also take pictures of different pages, combine them to make a single pdf file.*
5. *The name of the file should be <RollNumber>_LT2B. Ensure your roll number in the filename has only digits and uppercase characters.*
6. *Upload/attach your file. Make sure you click on the 'Turn in' button to submit your file.*
7. *11:30 PM is a strict deadline after which no submissions will be allowed.*
8. *We will not accept submission by any other means.*

1. What is the output of the following C program when the input is the first four characters of your first name.

```
#include <stdio.h>
int main () {
      char s[10], *p[10], *temp ;
      int i;

      scanf("%s",s); printf("%s\n",s);

      for (i=0; i<10; i++) p[i] = s+i;

      printf ("p[0] = %s,  p[1] = %s\n", p[0], p[1]) ;

      temp = p[0]; p[0] = p[2]; p[1] = temp;
      printf ("p[0] = %s,  p[1] = %s\n", p[0], p[1]) ;

      *p[4] = 'r';
      *p[5] = '\0';
      printf ("s = %s, p[0] = %s\n", s, p[0]) ;

}
```

[6 marks]

Solution:

For input `abcd` the output will be:

```
abcd
```

```
p[0] = abcd,  p[1] = bcd          2 marks
p[0] = cd,  p[1] = abcd           2 marks
s = abcdr, p[0] = cdr             2 marks
```

2. A string, w, may be converted to a palindrome by concatenating its reverse, w^R, with it. For example, the string `"no"` when concatenated by its reverse `"on"` creates the palindrome `"noon"`.
   (a) Complete the following C function by filling the blanks indexed by the red letters with suitable code so that the program prints the palindrome of the word read in the string a.
   (b) Write your first name. Suppose the input provided to the following program is your first name. Then answer the following questions:
      I. How many times is the function called, including the one from `main()`?
      II. What is the value of the variable, `i`, just before the function returns to `main()`?
      III. For which values of the variable, `i`, is the statement `s[i] = '\0'` executed?

```
#include <stdio.h>

void make_palindrome(___A___)
{
      ___B___ i=0;
      int temp;
      if (s[i] == '\0') return;
      temp = s[i]; i++;
      make_palindrome(s);
      s[i] = temp; ___C___; s[i] = '\0';
}

int main () {
      char a[100];
      scanf("%s", a);
      make_palindrome(a);
      printf("%s\n",a);
}
```

[6 marks]

Solution:
1 mark for each sub-answer.
(a)

```
void make_palindrome(char *s)
{
    static int i=0;
    int temp;
    if (s[i] == NULL) return;
    temp = s[i]; i++;
    make_palindrome(s);
    s[i] = temp; i++; s[i] = '\0';
}
```

(b) Suppose the name is Prakash, that is, a 7-letter name. Let N=7.
I. The function is called 8 times, that is N+1 times.

II. The value of i before it returns to main() is 14, that is 2N.
III. The statement s[i] = '\0' is executed for i = 8, 9, … , 14, that is, N+1, …, 2N

3. Consider an array **`int a[15][15]`**, whose elements are **`a[i][j] = 10*i+j`**. Let *p* and *q* be the last two digits of your roll number. For example, if your roll number is 20CS30075, then *p*=7 and *q*=5. Then what will be printed due to the following function call when *p* and *q* are the digits extracted from your roll number? Write your roll number at the beginning of your answer.

```
what((int(*)[15])&a[p+3][q+3],3,3);
```

where the function definition is as follows.

```
void what(int x[][15], int r, int c){
      int i, j;
      for(i=0; i > -r; --i){
            for(j=0; j > -c; --j) printf("%d ", x[i][j]);
            putchar('\n');
      }
}
```

[6 marks]

Solution:

Let the roll number be 75, that is pq. The following numbers are printed:

```
108 107 106           (p+3)(q+3)   (p+3)(q+2)   (p+3)(q+1)
          98 97 96                 (p+2)(q+3)   (p+2)(q+2)   (p+2)(q+1)
          88 87 86                 (p+1)(q+3)   (p+1)(q+2)   (p+1)(q+1)
```

Test code:

```
#include <stdio.h>
void what(int x[][15], int r, int c){
      int i, j;
            for(i=0; i > -r; --i){
                  for(j=0; j > -c; --j) printf("%d ", x[i][j]);
                        putchar('\n');
            }
}

int main () {
int a[15][15];
int i, j, p, q;

scanf("%d%d",&p, &q);
for (i=0; i<15; i++) for (j=0; j<15; j++) a[i][j] = 10*i+j;
    what((int (*)[15])&a[p+3][q+3], 3, 3);
}
```

4. A data type Student is defined as follows.

```
struct Student {
      char rollNo[10];
      float marks;
      int rank;
};
struct Student *result;
```

A file `input.dat` contains the number of student records, N, in the first line. Each of the following N lines contains a roll number and marks separated by a space. Your program should first read N, allocate memory to store N records in the array result, then read the N records from the input file and store them in `result[]`. The program should then calculate the rank of all the students (based on their marks) such that the highest mark is assigned rank 1, the second highest mark is assigned rank 2, and so on. Note that if two or more students are having the same marks then their rank will be same. For an example, consider the data given below.

Roll No.	R1	R2	R3	R4	R5	R6
Marks	65	90	75	80	65	75
Rank	4	1	3	2	4	3

You have to write a program to do the following:

- To assign rank against each student.
- Print the roll number(s) of the student(s) given a rank as input.

A program structure is given below. Complete the functions `main`, `read`, `sort`, `assign` and `search` so that the program executes to give the desired output. (The missing code should come right after the commented sections.)

```
struct Student {
      char rollNo[10];
      float marks;
      int rank;
};

int main() {
      FILE *fp;
      int N, r;
      struct Student *result;
      /* write code to open the input file; read N from the file;
            allocate memory for result; read student records into
            result[]*/
      ...
```

```
        sort(result,N);
        void assign(result,N);
        /* write code to read r */
        ...
        search(result,N,r);
        return 0;
    }

    void sort(struct Student *record, int n) {
        /* write code to sort the array in descending order of the
           marks */
        ...
    }

    void rank(struct Student *record, int n) {
        /* write code to rank each element in the array */
        ...
    }

    void search(struct Student *record, int n, int r) {
        /* write code to search the array record[] for entries with
        the rank value r and print the roll numbers corresponding to
        them (to stdout) */
        ...
    }
```

[16 marks]

Solution:

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

        struct Student {
          char rollNo[10];
          float marks;
          int rank;
```

```c
};

void sort(struct Student *record, int n) {
  int i,j;
  float t;
  char s[10];
  for(i=1; i<n; ++i){
    t = record[i].marks;
    strcpy(s,record[i].rollNo);
    j=i-1;
    while(j>=0 && record[j].marks < t){
record[j+1].marks = record[j].marks;
strcpy(record[j+1].rollNo, record[j].rollNo);
--j;
    }
    record[j+1].marks = t;
    strcpy(record[j+1].rollNo,s);
  }
}

void rank(struct Student *record, int n) {
  int i, r;
  float m;

  i=0; r = 1;
  while(i<n){
    m = record[i].marks;
    while(record[i].marks == m){
      record[i].rank = r;
      ++i;
    }
    ++r;
  }
}
```

```
void search(struct Student *record, int n, int r) {
  int i = 0;
  while(i<n){
    if(record[i].rank == r)
      break;
    ++i;
  }
  if(i == n)
    printf("No student has rank %d.\n", r);
  else{
    printf("Roll numbers of students with rank %d: \n",r);
    while(record[i].rank == r){
      printf("   %s\n",record[i].rollNo);
      ++i;
    }
  }
}

int main() {
  FILE *fp;
  int N, r;

  struct Student *result;

  if((fp = fopen("input.dat","r")) == NULL){
    printf("Error opening the file.\n");
    exit(0);
  }

  fscanf(fp, "%d", &N);
  result = (struct Student *)malloc(N*sizeof(struct Student));
```

```
    int i;
    for(i=0; i<N; ++i){
      fscanf(fp,"%s %f",result[i].rollNo, &(result[i].marks));
    }

    for(i=0; i<N; ++i){
      printf("   %s %f\n", result[i].rollNo, result[i].marks);
    }

    sort(result,N);

    printf("\nAfter Sorting...\n\n");
    for(i=0; i<N; ++i){
      printf("   %s %f\n", result[i].rollNo, result[i].marks);
    }

    rank(result,N);

    printf("\nAfter Ranking...\n\n");
    for(i=0; i<N; ++i){
      printf("   %s %f %d\n", result[i].rollNo, result[i].marks,
  result[i].rank);
    }

    printf("Enter a rank: ");
    scanf("%d", &r);
    search(result,N,r);

    return 0;
  }
```

5. There are two bags – one containing 50 red dice and the other containing 50 green dice. One picks nR (<=50) red dice from bag 1 and nG (<=50) green dice from bag 2 (nR+nG dice in total) in some order. Your task is to list all possible orders in which one can pick the dice. Let the character ‘r’ represent a red die and ‘g’ represent a green die. A possible order to pick the dice can be represented by a string of length nR+nG consisting of nR occurrences of ‘r’ and nG occurrences of ‘g’. For example, suppose that nR=3 and nG=2. Then the string “rgrrg” indicates that you pick a red die

first, then a green die, then 2 red dice followed by a green die. There are 10 different orders in which 3 red dice and 2 green dice can be picked - given by the strings "rrrgg", "rrgrg", "rrggr", "rgrrg", "rgrgr", "rggrr", "grrrg", "grrgr", "grgrr" and "ggrrr". Observe that the total number of orders is given by (nR+nG)!/(nR! nG!) where x! denotes factorial of x.

Write a C program that, on input two positive integers nR and nG, prints all possible ways to pick nR red dice and nG green dice. The output should contain strings, each printed in a separate line, representing all possible orders (without repetitions). After reading nR and nG, your `main()` program should allocate memory for a char type pointer str to store a null-terminated string of nR+nG symbols. Pass str, nR, nG and anything else that may be necessary as arguments to a function **`print_orders(...)`** that **recursively** computes and stores the different possible orders in str. The function must also print `str` when it is null-terminated.

```
Sample Intput/Output:


   2 5
   nR = 2, nG = 5


   rrggggg
   rgrgggg
   rggrggg
   rgggrgg
   rggggrg
   rgggggr
   grrgggg
   grgrggg
   grggrgg
   grgggrg
   grggggr
   ggrrggg
   ggrgrgg
   ggrggrg
   ggrgggr
   gggrrgg
   gggrgrg
   gggrggr
   ggggrrg
   ggggrgr
   gggggrr
```

[16 marks]

Sample Solution:

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

```
void print_orders(int m, int n, int curri, char *s){
  if (n == 0 && m == 0){
     s[curri] = '\0';
     printf("%s\n",s);
     return;
  }
  if(n == 0){
     s[curri] = 'r';
     print_orders(m-1,0, curri+1, s);
     return;
   }
   if(m == 0){
     s[curri] = 'g';
     print_orders(0,n-1, curri+1,s);
     return;
   }
   s[curri] = 'g';
   print_orders(m, n-1, curri+1,s);
   s[curri] = 'r';
   print_orders(m-1, n, curri+1,s);
}

int main(){
   int nR, nG;
   char *str;
   printf("\n");
   scanf("%d %d", &nR, &nG);
   if (nR<1 || nG<1){
      printf("Invalid! Enter positive integers only.\n\n");
      return 1;
   }
   printf("nR = %d, nG = %d\n\n",nR,nG);
   str = (char*)malloc((nR+nG+1)*sizeof(char));
   print_orders(nR,nG,0,str);
   printf("\n");
   return 0;
}
```