Indian Institute of Technology Kharagpur
Department of Computer Science and Engineering

CS10003 Programming and Data Structures

Spring Semester, 2021-2022 **Sections 9 & 10**

Long Test 1 – Part B[Marks = 50] **18-May-2022, 10:15 to 11:25**

INSTRUCTIONS

- You have 1 hour for writing and 10 minutes for submission.
- Write your answers on paper. Answers must be handwritten. Typed or written answers using an electronic device are not allowed.
- Write your name and roll number on each page. Write page numbers for each page.
- Scan all pages and collate. Create a single PDF file (of size $\leq$ 10 MB). You could also take pictures of different pages, combine them to make a single pdf file.
- The name of the file for this part should be <RollNumber>_Long1B. Ensure your roll number in the filename has only digits and uppercase characters.
- Upload your file. Make sure you click on the 'Turn in' button to submit your file.
- The said deadlines are strict after which no submissions will be allowed.
- We will not accept submission by any other means.

1. Write a program that on the input of a string w consisting of only letters, separates the lowercase and uppercase letters. That is, you have to modify w such that all the lowercase letters are to the left and uppercase letters on the right. The order of the letters need not be retained. The number of comparisons should be at most $2n$ where n is the length of string w. Assume that the length of the input string is at most 49. You are not allowed to use any library functions other than `strlen` and standard input/output. Your program should have only the *main*() function. [12]

Sample Output

```
Enter string: dYfJlslTwXKLp
Modified string: dpfwlslTXLKJY
```

Solution:

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

int main(){
  char w[50], t;
  int i, j;
  printf("Enter string: ");
  scanf("%s", w);

  i = 0;
```

```
  j = strlen(w) - 1;
  while(i < j){
    if(w[i] >= 'A' && w[i] <= 'Z'){
      t = w[i];
      w[i] = w[j];
      w[j] = t;
      --j;
    }
    else
      ++i;
    if(w[j] >= 'a' && w[j] <= 'z'){
      t = w[i];
      w[i] = w[j];
      w[j] = t;
      ++i;
    }
    else
      --j;
  }
  printf("Modified string: %s, %d\n",w);
  return 0;
}
```

2. Let $A[]$ be an unsorted array of n positive integers. We want to print the elements of $A[]$ in the ascending order. Assume that the elements of $A[]$ are distinct from one another. Complete the following code by filling in the blanks. After we print an element of $A[]$, we replace that element by –1 so that this element is not considered again for minimum calculations in future. You are not permitted to use any variables other than what is defined already. 8

```
void listall ( int A[], int n )
{
  int i, j;
  int minidx;

  for (i=0; i<n; ++i) {
     /* Find the index of the smallest unprinted element */
     minidx = -1 ;
     for (____________________) { /* Loop on j */            (A)
       if (___________________________________) minidx = j;   (B)
     }
     printf("%d ", A[minidx]); /* Print the minimum */
     A[minidx] = ___________________ ;                       (C)
  }
  printf("\n");
}
```

Solution:

```
(A) j=0; j<n; ++j
(B) (A[j] >= 0) && ((minidx == -1) || (A[j] < A[minidx]))
(C) -1
```

3. In a compounded fixed deposit investment, a one-time principal amount P is invested for N years with interest computed yearly at R percent per annum. The interest $I(i) = A(i-1)(r/100)$ (earned at the end of the i-th year) on the amount $A(i-1)$ (accrued at the beginning of the year i, $i = 1, 2, ..., N$), is added to $A(i-1)$ to get amount $A(i)$ (accrued at the end of the i-th

year), that is, $A(i) = A(i-1) + I(i) = A(i-1) + A(i-1)(r/100)$. So, $A(0) = P$ and $A(1) = A(0) + I(1) = A(0) + A(0)r/100 = P(1 + r/100)$ and $A(2) = A(1) + I(2) = A(1) + A(1)(r/100)$ and so on.

(a) Write a C function simpleinterest with prototype float simpleinterest(float Principal, float R) to compute and return $I(i)$ given Principal=$A(i-1)$ at rate of interest R percent per annum, as inputs.

(b) Write another C function with prototype float compoundinterest(float Principal, float R, int N) that computes and returns $A(N)$, given $Principal = A(0) = P$, the interest rate R and the number of years N, as inputs. This function must repeatedly use the above defined function simpleinterest N times.

An investor invests an amount P in the beginning of every year i, $i = 1, 2, 3, N$ for N consecutive years, for earning fixed deposit interest for the next $N - i + 1$ years.

(c) Write the main C program for computing the accrued amount at the end of N years using the function compoundinterest above N times, viewing the total accrued amount as a sum of N fixed deposits of principal amounts P, for varying periods $N, N-1, ..., 1$ years, respectively.

You must not use any math.h functions. 10

Solution:

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

float simpleinterest(float principal,float r);

float compoundinterest(float principal,float r,int n);

void main(){
  float r,factor,amount,simpleint,camount,interest,yly;
  int n;

  printf("Enter number of years, and annual rate of interest in percentage, yearly recurring deposit:" );
  scanf("%d %f %f",&n,&r,&yly);


  int i;
  float totalfds=0;
  for (i=1;i<=n;i++){
    totalfds=totalfds+compoundinterest(yly,r,n-i+1);
  }
  printf("rd accrued = %f\n",totalfds);
}

float simpleinterest(float principal,float r) {
   float simpleint=principal*r/100;
   return(simpleint);
}

float compoundinterest(float p,float r, int n) {
  float camount=p; int i;
  for (i=1;i<=n;i++) {
    camount=camount+simpleinterest(camount,r);
  }
  return(camount)
}
```

4. Write a C function that takes as arguments three integer arrays, A, B, C along with integers m, n indicating the number of elements in A and B, respectively. The arrays A is assumed to be sorted in ascending order and B is assumed to be sorted in descending order. You are required to store in C all elements that are present in both A and B, in ascending order. You may assume that A and B individually may have duplicate elements within them. In the result, there should not be any duplicates in C. The function should return the number of elements in C. For example, if $A = \{8, 8, 12, 12, 15, 67\}$ and $B = \{88, 67, 67, 45, 15, 12, 12, 9, 1\}$ with $m = 6$, $n = 9$, the resulting C should be $\{12, 15, 67\}$ and 3 should be returned. Do not use any additional arrays or any library functions other than standard input and output. Write only the required function. No need to write the main function. 12

Solution:

```
#include<Stdio.h>
int merge_new(int A[], int B[], int C[], int m, int n)
{
    int p, i, j;
    p = 0;
    for (i = 0, j = n-1; ((i < m)&& (j < n)); )
        {
            if (A[i] == A[i+1]) i++;
            if (B[j] == B[j-1]) j--;
            if (A[i] == B[j])
                {
                    C[p++] = A[i];
                    i++;
                    j--;
                    continue;
                }
             if (A[i] < B[j]) i++;
             else j--;
        }
        return(p);
}

main()
{
    int A[20], B[20], C[20];
    int m, n, p, i;
    scanf("%d%d", &m, &n);
    printf("m = %d, n = %d \n", m, n);
    for(i=0; i< m; i++) scanf("%d", &A[i]);
    printf("A = ");
    for(i=0; i< m; i++) printf("%d, ", A[i]);
    printf("\n");
    for(i=0; i< n; i++) scanf("%d", &B[i]);
    printf("B = ");
    for(i=0; i< n; i++) printf("%d, ", B[i]);
    printf("\n");
    p = merge_new(A, B, C, m, n);
    printf("p = %d \n", p);
    printf("C = ");
    for(i=0; i< p; i++) printf("%d, ", C[i]);
    printf("\n");
}
```

5. Consider the following recurrence relation: $5f(n) = f(n+1) + 6f(n-1)$, where $f(0) = 0$, $f(1) = 1$. Write a recursive function to compute $f(n)$. 8

Solution:

```
#include<stdio.h>

int f(int n) {
  if (n==0)
    return 0;
  else if (n==1)
    return 1;
  else
    return 5*f(n-1)-6*f(n-2);
}

int main () {
  int n;
  scanf("%d",&n);
  printf("f(%d)=%d \n",n,f(n));

}
```