ESC101: Fundamentals of Computing(Major Quiz 2)

Set B

Instructions:

1. Write you name, section and roll number on all the pages of the answer book.
2. Write the answers cleanly in the space provided. There is space left on the back of the answer book for rough work.
3. Using pens (blue/black ink) and not pencils. Do not use red pens for answering.
4. Even if no answers are written, the answer book has to be returned back with name and roll number written.
5. Recall that cheating carries severe consequences.

Question	Points	Score
1	10	
2	40	
3	50	
Total:	100	

Name:	
Roll No:	
Section:	

Name: Section: Rollno:

Question 1. (10 points) Answer the following short questions. An ASCII value table has been provided at the end of question paper. For each correct answer 5 marks are allotted.

#	Question	Answer

A.	`char arr[] = "Solar Orbit";`
	`char *s;`
	`s+=3;`
	`s=arr;`
	`s+=3;`
	`*s=100;`
	`printf("%s",arr);`
		Soldr Orbit

B.	`struct node{`
	`char x,y;`
	`};`
	`int main()`
	`{`
	`struct node n1={'c','z'};`
	`struct node n2={ 'a','d'};`
	`printf("%d %d",n1.x-n2.x,n1.y-n2.y);`
	`}`
		2 22

Name: Section: Rollno:

Question 2. (40 points) Consider the program given below.

```
#include <stdio.h>

void copy(int F[2][2],int M[2][2]){
        M[0][0] = F[0][0];
        M[0][1] = F[0][1];
        M[1][0] = F[1][0];
        M[1][1] = F[1][1];
}

void matrixFunction(int F[2][2], int M[2][2]){
  int x =  F[0][0]*M[0][0] + F[0][1]*M[1][0];
  int y =  F[0][0]*M[0][1] + F[0][1]*M[1][1];
  int z =  F[1][0]*M[0][0] + F[1][1]*M[1][0];
  int w =  F[1][0]*M[0][1] + F[1][1]*M[1][1];
  F[0][0] = x;
  F[0][1] = y;
  F[1][0] = z;
  F[1][1] = w;
}

void matrixRecursion(int F[2][2], int n){
        int identity[2][2]={{1,0},{0,1}};
        if(n==0){
                F[0][0]=1;
                F[0][1]=0;
                F[1][0]=0;
                F[1][1]=1;
        }else{
                int M[2][2];
                copy(F,M);
                matrixRecursion(F,n/2);
                matrixFunction(F,F);
                if(n%2==1){
                        matrixFunction(F,M);
                }
        }
}

int main(){
        int n;
        scanf("%d",&n);
        int ans = 0;
        if(n>=1){
                int F[2][2] = {{1,1},{1,0}};
                matrixRecursion(F,n-1);
                ans = F[0][0];
        }
        printf("%d\n",ans);
        return 0;
}
```

Name: Section: Rollno:

What is the output of the program from the following inputs. For each testcase 8 marks are allotted.

Input:
`11`
Output:

Solution:
89

Input:
`8`
Output:

Solution:
21

Input:
`1`
Output:

Solution:
1

Input:
`16`
Output:

Solution:
987

Input:
`6`
Output:

Solution:
8

BONUS QUESTION (2 points): Identify what the program does.

Solution:
Outputs the n^{th} fibonacci number.

Name: Section: Rollno:

Question 3. (50 points) Consider the program given below.

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

void print_arr(int*arr, int size);

int findMedian(int * arr, int s, int e)
{
    /* This function returns the index of the median in the array
       between the indices s and e (inclusive)

          For Exmaple, if  arr=1,3,4,2,5
          And we call the function findMedian(arr,1,3)
          It Returns 1 (As the index of 3 in arr is 1)

          If the range (s to e) has even number of integers, then the
             function returns the index of the smaller of the two
             middlemost elements.
          For Exmaple, if  arr=1,3,4,2,5
          And we call the function findMedian(arr,0,3)
          It Returns 3 (As the index of 2 in arr is 3), 2 being the
             smaller of the two middlemost elements in this case.
   */
}
int Partition(int * X, int s, int e)
{
    int pivotIndex=findMedian(X,s,e);
    int temp=X[pivotIndex];
    int midIndex=(e+s+1)/2;
    X[pivotIndex]=X[midIndex];
    X[midIndex]=temp;
    int leftIndex=s;
    int rightIndex=midIndex+1;
    while(leftIndex<midIndex && rightIndex<=e)
    {
        if(X[rightIndex]>=X[midIndex])
        {
            rightIndex++;
        }
        else
        {
            int temp2=X[rightIndex];
            X[rightIndex]=X[leftIndex];
            X[leftIndex]=temp2;
            leftIndex++;
        }

    }
    return midIndex;
}

```

```
47 void quickSort(int * X,int s ,int e)
48 {
49     if(e-s<=0)
50     {
51         return;
52     }
53     int pivot=Partition(X, s, e);
54     quickSort(X,s,pivot-1);
55     quickSort(X,pivot+1,e);
56 }
57
58 int main()
59 {
60         int N;
61         scanf("%d",&N);
62         int *X;
63     X=(int*)malloc(sizeof(int)*N);
64
65         for(int i=0;i<N;i++)
66         {
67         scanf("%d",&X[i]);
68         }
69     quickSort(X,0,N-1);
70     print_arr(X,N);
71         return 0;
72 }
73
74 void print_arr(int *arr, int size)
75 {
76     printf("\n");
77     for(int i=0;i<size;i++)
78         {
79         printf("%d ",arr[i]);
80     }
81     printf("\n");
82
83
84 }
```

What is the output of the program? If the program results in an error, mention the type of error. For each testcase 10 marks are allotted.

Input:
3
5 4 3
Output:

Solution:
3 4 5

Input:
4
4 2 3 1
Output:

Solution:
3 1 2 4

Input:
5
8 7 4 5 2
Output:

Solution:
4 2 5 8 7

Input:
5
9 8 6 5 7
Output:

Solution:
6 5 7 9 8

Input:
5
1 5 2 7 9
Output:

Solution:
2 1 5 9 7

Name: Section: Rollno:

Characters	ASCII Values
A – Z	65 – 90
a – z	97 – 122
0 – 9	48 – 57
special symbols	0 - 47, 58 - 64, 91 - 96, 123 - 127