SRN | | | | | | | | | | | | |

PES University, Bengaluru

(Established under Karnataka Act No. 16 of 2013)

UE21CS141B

JULY 2022: END SEMESTER ASSESSMENT B.TECH II SEMESTER

UE21CS141B – PROBLEM SOLVING WITH C

Time: 3 Hrs | Answer All Questions | Max Marks: 100

Q1 a) Explain different options of gcc to get the output of C code. — 4 M

Solution: Any two options, 4 marks

gcc –c filename.c // preprocess and compile
gcc filename.o linking
a.exe OR ./a.out

gcc filename.c // preprocess, compile and link
a.exe OR ./a.out

gcc –E filename.c // output of preprocessing on the terminal / screen
Can be –o, OR –save-temps

b) The data file mobiles.txt has the details (**Model_id, Price in Rs and the Number of quantities available**) of 1000 Mobiles. Sample data is given below. A person named Hrithik would like to buy mobile for himself and checking the availability of mobile phones. Write the C code to display the IDs of Mobile phones which are not available (quantity is NIL). — 6 M

Every column in a row is separated by a tab in the data file. Use **redirection operator** to provide the data file as an input to the executable.

Hint: a.exe < mobiles.txt OR ./a.out < mobiles.txt

1	**6499**	**650**
10	**7999**	**80**
3	**8999**	**707**
6	**11990**	**0**
5	**10990**	**1099**
4	**9990**	**676**
7	**19990**	**1999**
8	**36990**	**0**
9	**36990**	**3699**
2	**25990**	**2599**

…

Solution:

```
#include<stdio.h>
int main()
{
        int i = 0;      int id;   int price;  int num_quant; // variable declarations 1 mark
        while( i<= 1000)                                    // any loop, 1 mark
        {       scanf("%d %d %d",&id,&price,&num_quant);   // 1 mark
                if( num_quant == 0)             //  1 mark
                {       printf("Model id is %d\n",id); // 1 mark              }
                i++;  // 1 mark
        }
        return 0;
}
```

SRN | | | | | | | | | | | | |

	c)	What does the below code do? List any two variables in the same code. #include<stdio.h> int main() { int i; int j,sum; for(i = 0, j = 0;i<100;i++,j++) { sum = i*j; } printf("%d",sum); return 0; } **Solution:** The code finds the square of all the numbers from o to 99 and stores it in sum each time. Prints only the last square outside. // 2mark Variables: i, j, sum – Any two, 2marks	4 M
	d)	Explain do-while and switch constructs in C with a coding snippet **Solution:** 3 marks each //description : 1 mark // code: 2 mark do –while loop: The body of do...while loop is executed at least once. For loop to continue, the test expression is evaluated for TRUE; terminated if the evaluation results in FALSE. do { for(s=0; n; n/2) { s+=n%10 } n=s; }while(s>9); Switch: selection statement Switch is used in following cases • We are comparing integers (integral values) • We are comparing a variable(an expression) with constants • We are comparing for equality (no > or <) • In all comparisons, the same variable is used. switch(count) { case 0: printf("scalene\n"); break; case 3: printf("equi\n"); break; case 1: printf("iso\n"); break; }	6 M **(3+3)**
Q2	a)	Provide a brief note on below commands in gdb. i) info ii) print **Solution: 2marks each** Info : To print a list of all breakpoints, watchpoints, and catchpoints Info b OR info break print: It evaluates and prints the value of an expression of the language your program is written in. p expression: prints the value of a given expression. p variable: prints the value of a given variable. p function_name :: variable: To specify a static variable in a particular function or file colon-colon (::) notation is used.	4 M **(2+2)**
	b)	Given the client code, write the implementations for Read_arrays and Display_sum_arrays. Read_arrays function must read n elements from the user and store it in arr. Display_sum_arrays function must find the sum of all the elements in arr and display the result.	6 M

		n is the number of elements to be entered by the user. #include<stdio.h> void Display_sum_arrays(int *a,int n); void Read_arrays(int *a, int n); int main() { int arr[1000]; int n; scanf("%d",&n); Read_arrays(arr, n); Display_sum_arrays(arr, n); return 0; } **Solution:** void Display_sum_arrays(int *a,int n) { int sum = 0; // 1 mark for(int i = 0 ; I < n; i++) // 1 mark { sum += a[i]; // 1 mark } printf("%d",sum); // 1 mark } void Read_arrays(int *a, int n) { for(int i = 0 ; I < n; i++) // 1 mark for loop { printf("enter the element\n"); scanf("%d",&a[i]); // 1 mark for proper scanf with & } }	
	c)	Find the output of below code snippet when executed separately. i) char a[] = "Health is wealth"; a[1] = 'E'; printf("%s",a); ii) int a[ ] = {7,4,9,0,12,87}; printf("%d %d", a[6], a[5]); **Solution: // 2 marks each** i) HEalth is wealth ii) undefined behaviour 87	4 M **(2+2)**
	d)	Write the user defined function to copy one string to another string. Also test this function in the client code. **Solution: // 4 marks for implementation – might include arrays and pointer notations, recursive functions – All okay** **// 2 marks for client code** void my_strcpy(char *b, char *a) //4 marks { // copy a to b int i =0; while(a[i] != '\0') // 1 mark { b[i]=a[i]; i++; // 1 mark } b[i] = '\0'; // append '\0' at the end // 1 mark } int main() // 2 marks { char mystr1[] = "pes university"; char mystr2[100]; printf("%s\n",mystr1); my_strcpy(mystr2, mystr2); printf("%s\n",mystr2); }	6 M
Q3	a)	What is Dangling pointer? Explain with a code snippet. **Solution:**	4 M

SRN | | | | | | | | | | | | |

Pointer which points to a location that doesn't exist -- // 1 mark
int *p = (int*) malloc(sizeof(5*sizeof(int)));
*p = 80;
free(p); // p becomes dangling // 1 mark
diagram // 2 marks

b) Complete the function definition of extract_data_display to segregate the even and odd numbers from the given linked list and copy those elements to respective arrays. The function must also print both the arrays. The client code is as below. — 6 M

```
#include<stdio.h>
#include<stdlib.h>
struct node
{       int data;       struct node* link;    };
typedef struct node NODE;
struct list
{       NODE *head;    };
typedef struct list LIST;
void extract_data_display(LIST* li,int *even,int *odd);
int main()
{
        NODE *n = (NODE*) malloc(sizeof(NODE));
        n->data = 40;           n->link = (NODE*) malloc(sizeof(NODE));
        n->link->data = 33;     n->link->link = (NODE*) malloc(sizeof(NODE));
        n->link->link->data = 25;       n->link->link->link = (NODE*) malloc(sizeof(NODE));
        n->link->link->link->data = 88;  n->link->link->link->link = NULL;
        LIST *li;       li->head = n;   int odd[100];    int even[100];
        extract_data_display(li,even,odd);
        return 0;
}
void extract_data_display(LIST *li,int *even,int *odd)
{   // Fill this implementation     }
```

Solution: 1 mark each

```
void extract_data_display(LIST *li,int *even,int *odd)
{
int i=0,j=0;
if(li->head == NULL)
printf("no elements in the list to display\n");
else
{
NODE *n = li->head;
while(n != NULL) // loop 1 mark
{
if(n->data % 2 == 0) // 1 mark
{
even[i] = n->data; // 1 mark
i++; // 1 mark
}
else
```

		{ odd[j] = n->data; j++; } n = n->link; } int k; printf("Even elements in the list are\n"); // printing both the arrays 2 marks for(k = 0;k < i;k++) printf("%d\t",even[k]); printf("\n"); printf("Odd elements in the list are\n"); for(k = 0;k < j;k++) printf("%d\t",odd[k]); printf("\n"); } }	
	c)	What gets printed? #include<stdio.h> struct Example { int a; int *c; }; int main() { struct Example e1,e2; int b = 48; e1.a = 28; e1.c = &b; e2 = e1; printf("%d ", *(e2.c)); *(e2.c) = e2.a; printf("%d ",*(e1.c)); return 0; } **Solution: // 2 marks each** 48 28	4 M
	d)	There is a structure called student and two data members in it such as name and age. Write the code to do the following. i) Create a structure variable s ii) Create a pointer to structure variable which points to s iii) Display name using pointer iv) Display age using s v) Create a structure variable s1 and copy the contents of s to s1.Write the code snippet to compare s1 and s **Solution: 1 mark each** i) struct student s; ii) struct student *p = &s; iii) printf("%s",p->name); iv) printf("%d",s.age); v) struct student s1 = s; // 1 mark if(s1.age == s.age && strcmp(s1.name, s.name) == 0) // 1 mark printf("structures are equal") else printf("not same")	6 M
Q4	a)	Below code must write the details from age and name arrays to a data file named emp.txt. Data from the same indices must be copied to a file in one row separated by a space. Fill up the blank spaces to do the same.	4 M

SRN | | | | | | | | | | | | |

```
#include<stdio.h>
int main()
{
        int age[ ] = {34,45,32,54,44};
       char name[][50] = {"raj","rajesh","anil", "anitha","rajendra"};
       FILE *fp = fopen("emp.txt","w");
       if(_________________)
       {
           for(int i = 0 ; i < 5; i++)
           {
                  fprintf(fp,"___________ \n",age[i],name[i]);
           }
       }
      fclose(fp);
      return 0;
}
```

Solution: // each 2 marks

fp != NULL **or** fp

%d %s

b) Implement Binary search using recursive method on an array of 100 integer elements which are stored in ascending order. Handle both successful and unsuccessful search. — 6 M

Given the array, int a[ ] = {100,98,76,54,44,43,42,40,31,30};

Write only the function definition. Client code is not a requirement

Solution:

```
int mysearch(int a[],int low,int high,int key)
{
          if(low > high)      // base condition   1 mark
                    return -1;
           else
           {
                    int mid = (low+high)/2;   // 1 mark
                    if(a[mid]==key)
                    {
                              return mid;  // 1 mark
                    }
                    else if(key<a[mid])
                              return mysearch(a,low,mid-1,key);  // 1 mark
                    else
                              return mysearch(a,mid+1,high,key); // 1mark
           }
}
```

c) Explain array of pointers to integers with a code snippet. — 4 M

Solution:

```
#include<stdio.h>
int main()
{
          int data[] = {3,8,1,4};
          int *pdata[4];
          for (int i = 0;  i < 4; i++)
                pdata[i] = &data[i];   // 2 marks
          for (int i = 0;  i < 4; i++)
                printf("%d",*(pdata[i]));  // 2 marks
```

SRN | | | | | | | | | | | | |

```
        return 0;
}
```

d) Given a file stores.txt containing storeid (string), store area code (string), items available in the store(integer), number of customers (integer) and store sales (integer). The sample is as below. 6 M

```
9       1090    1321    680     46310
10      1030    1235    1130    44150
38      1174    1411    1080    62870
42      965     1152    600     48140
45      1009    1194    520     35800
53      1074    1288    320     70450
72      1152    1380    530     33580
73      891     1073    630     67370
101     1096    1321    900     78420
111     1162    1382    1260    51700
117     1157    1379    770     80780
125     1164    1390    370     35510
```

Sort it based on the number of customers and based on store sales separately using selection sort algorithm.

The new type struct Store is used to store the details of each store. The array store_arr contains the data from the dataset. Sort the data in an array of structures as per the requirement in the client code. n is the number of store details from the file.

Include the definitions of compare_sales and compare_customers.

```
struct Store
{       int store_id;   int num_customers;      int num_store_sales;        };
typedef struct Store STORE;
int main()
{
        STORE store_arr[10000];
        // Code to read the contents from the datafile and storing those details in the array of structure is available. Please DO NOT add that code here
        int ch;
        printf("enter the choice.\n1. sort on sales\n2. sort number of customers\n");
        scanf("%d",&ch);
        switch(ch)
        {       case 1: sort(store_arr, n, compare_sales); break;
                case 2: sort(store_arr, n, compare_customers); break;
                default: printf("exiting from the program"); break;
        }
        return 0;
}
void selection_sort(STORE *t, int n, int (*com)(STORE*,STORE*))
{       // fill this implementation             }
```

Solution:

```
void selection_sort(STORE *t, int n, int (*com) (STORE*,STORE*))
{
        int i,pos,j;
        for(i = 0;i<n-1;i++)                    // 2 loops 1 mark
        {
                pos = i;                        // 1 mark
                for(j = i+1;j<n;j++)
                {
                        if(com(t[pos],t[j]))            // 1 mark
```

SRN | | | | | | | | | | | |

```
                        {
                                pos = j;          // 1 mark
                        }
                    }
                    if(pos != i)
                    {                             // swap 1 mark
                       STORE temp = t[pos];
                        t[pos] = t[i];
                        t[i] = temp;
                    }
            }

}
```

// 1 mark for any one definition out of two functions

```
int compare_ sales(STORE s1, STORE s2)
{       return s1.num_store_sales > s2. num_store_sales;    }
int compare_customers(STORE s1, STORE s2)
        {           return s1.num_customers > s2. num_customers;    }
```

Q5 a) Write a program to accept integer values in the command line and display the product of those integers. **4M**

Solution:

```
# include<stdio.h>
int main (int argc, char *argv[])   // 1 mark
{       int i, res;
      res = 1;
      if (argc < 2)
      {       printf (“The name of the program is %s\n”, argv[0]);    }
      else
      {
              for (i = 1;i < argc; i ++)    // 2 marks loop with  argc and atoi
                    res = res* atoi(argv[i]);
              printf( “Product is %d\n”, res);  // 1 mark
      }
      return 0;
}
```

b) List out any four characteristics of macro in C. Also find the output of below code. **6 M (4+2)**

```
#include<stdio.h>
#define SUM(a,b) a*b
int main()
{
        printf("%d",SUM(5+2,2+1));          return 0;
}
```

Solution: // any four, 4 marks

Macro does not judge anything
No memory Allocation for Macros
Can define string using macros
Can define macro with expression
Can define macro with parameter
Macro can be used in another macro
Constants defined using #define cannot be changed using the assignment operator
Redefining the macro with #define is allowed. But not advisable

// Output: 10   // 2 marks

SRN | | | | | | | | | | | |

	c)	**Say True or False** i) Array of bit fields is allowed ii) Bit fields with a length of 0 must be unnamed iii) Accessing the Variable length Arguments from the function body makes use of macros available in stdarg.h iv) Storing the symbol of one enum in another enum variable is invalid in C. **Solution:** False True True False	4M
	d)	i) The below code results in #include<stdio.h> enum Sample { A, B=8, C, D }; int main() { enum Sample s = B; printf("%d\t%d\n", s+C, s+A); return 0; } ii) What is a storage class in C? Explain static with an example code snippet. **Solution:** i) 17 8 ii) definition – 1 mark static meaning – 1 mark any related code - 2 marks **Storage Classes are used to describe the features of a variable/function. These features basically include the scope(visibility) and life-time** which help us to trace the existence of a particular variable during the runtime of a program // 1 mark for scope and lifetime words **A static variable tells the compiler to persist the variable until the end of program**. Instead of creating and destroying a variable every time when it comes into and goes out of scope, static is initialized only once and remains into existence till the end of program static int j=30; int main( ) { f1(); f1(); f1(); return 0; } void f1() { int i=0; printf("%d %d\n",i,j); i++; j++; }	6 M **(2+4)**