

Your ID No.

Name:

1. Consider the following two pieces of code that appear in a program that uses an array of pointers to strings and a 2D array of strings to sort MAX names.

```
char temp[100];
for (i = 0; i < MAX-1; ++i)
    for (j = i+1; j < MAX; ++j)
        if (strcmp(arr1[i],arr1[j]) > 0)
        {
            strcpy(temp,arr1[i]);
            strcpy(arr1[i],arr1[j]);
            strcpy(arr1[j],temp);
        }
```

```
char *tmp;
for (i = 0; i < MAX-1; ++i)
    for (j = i+1; j < MAX; ++j)
        if (strcmp(arr2[i],arr2[j]) > 0)
        {
            tmp = arr2[i];
            arr2[i] = arr2[j];
            arr2[j] = tmp;
        }
```

- (i) arr1 cannot be the array of pointers to strings because: [1½]

strcpy() cannot be used to change string literals that are read-only, for the compiler decides where to store them at the time of initialization of the array of pointers.

- (ii) arr2 cannot be the 2D array of strings because: [1½]

An array type (e.g., arr2[i]) cannot be modified through an assignment statement.

- (iii) The name of the sorting algorithm used here is: [½]

Selection sort

2. In no more than two sentences, write what the following program accomplishes: [3]

```
int main()
{
    char word[50], output[50];
    int len = 0;
    scanf("%s",word);
    strcpy(output,word);
    while (strcmp(word,"*") != 0)
    {
        if (strlen(word) > len)
        {
            strcpy(output,word);
            len = strlen(word);
        }
        scanf("%s",word);
    }
    if (!strcmp(output,"*"))
        printf("No word was entered.\n");
    else
    {
        printf("\n%s\n",output);
        printf("%d\n",len);
    }
    return 0;
}
```

The program keeps taking words as input until a * is typed by the user, and prints out the longest word input by the user and its length, or a message if no word was

3. isPalind() has been written to check if its argument string is a palindrome or not. Complete it. [3]

```
int isPalind(char *s) { /* checks if the given string is a palindrome or not */
    char *end = s + strlen(s) - 1; /* pointing to the element before \0 */
    for ( ; s < end ; s++, end-- )
        if ( *s != *end ) /* characters are different */
            return 0;
    return 1; /* yes, palindrome! */
}
```

4. An airlines accept any one of these documents as ID proofs during checking in: (a) passport (b) Indian driving license (c) Aadhaar card. A passport number can contain both alphabet and numbers and are exactly 8 characters in length; a driving license number can contain alphabets, numbers and special characters totaling a maximum of 12 characters; and Aadhaar number contains exactly 12 digits. [3]

(i) Declare a user-defined data type that is most efficient to store the type of ID a passenger produces ('P' for passport, 'D' for driving license and 'A' for Aadhaar), as well as the ID number itself.

```
typedef struct id {           (Alternatively, the structure can be defined also without
    char type;               using typedef.)
    union {
        char passport[9];
        char license[13];
        long long int aadhaar;
    };
} IDPROOF;
```

(ii) Declare an array of 100 elements of this new data type.

```
IDPROOF arr[100];
(or)
struct id arr[100];
```

Study each of the following code snippets looking out for any possible errors. If you identify any, then **state the type of error** (compile-time/run-time/logical). Next, **write/modify exactly one line** of code that would rectify the error and produce a perfectly executable code. For those code fragments that are completely error-free, write down what output is produced. [5 x 1½ = 7½]

5. To convert all the lowercase letters into uppercase of the input string and print the modified string:

```
char line[100], *cpy = line;
scanf("%[^\\n]", line);
while (*cpy = toupper(*cpy)) cpy++;
printf("%s\\n", cpy);
```

Logical error.
printf("%s\\n", line);

6.

```
float *getMarks(int);
int main() {
    float *marks;
    int i;
    marks = getMarks(10);
    for (i = 0; i < 10; ++i)
        printf("%.2f\\n", marks[i]);
}
```

```
float *getMarks(int n)
{
    float data[n]; /* VLA */
    int i;
    for (i = 0; i < 10; ++i)
        scanf("%f", &data[i]);
    return data;
}
```

Runtime error. The VLA declaration inside the function must be replaced with:

```
float *data = (float *) calloc(n, sizeof(float)); (or)
float *data = (float *) malloc(n*sizeof(float));
```

7.

```
int *pi;
*pi = 40;
printf("%d\\n", *pi);
```

Runtime error. Corrected line should read: `int i, *pi = &i;`
(or)
`int *pi = (int *) malloc (sizeof(int));`

8.

```
typedef struct {
    int numer;
    int denom;
} FRAC;

int main()
```

```
{
    FRAC f1 = {10,13};
    FRAC *fp = &f1;
    printf("Fraction: %d / %d\\n", *fp.numer,
        *fp.denom);
}
```

Compile-time error. `printf("Fraction: %d / %d\\n", (*fp).numer, (*fp).denom);` (or)
`printf("Fraction: %d / %d\\n", fp->numer, fp->denom);`

9.

```
char months[][4] = {"Jan", "Feb", "Mar"};
1[months][2] = 'D';
for (int i = 0; i < 3; ++i)
    printf("%s\\n", months[i]);
```

Error-free code that produces this output:
Jan
FeD
Mar