```c
// Q1: malloc() - Single Integer
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *ptr;
    ptr = (int *)malloc(sizeof(int));
    if (ptr == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }
    printf("Enter an integer value: ");
    scanf("%d", ptr);
    printf("You entered: %d\n", *ptr);
    free(ptr);
    return 0;
}

// Q2: calloc() - Store and Print n Integers
#include <stdio.h>
```

```c
#include <stdlib.h>

int main_calloc() {
    int *ptr, n, i;
    printf("Enter number of integers: ");
    scanf("%d", &n);
    ptr = (int *)calloc(n, sizeof(int));
    if (ptr == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }
    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) scanf("%d", &ptr[i]);
    printf("Stored integers are:\n");
    for (i = 0; i < n; i++) printf("%d ", ptr[i]);
    printf("\n");
    free(ptr);
    return 0;
}
```

```
// Q3: malloc() - Sum and Average of n Numbers
#include <stdio.h>
#include <stdlib.h>

int main_sum_avg() {
    float *ptr, sum = 0, average;
    int n, i;
    printf("Enter how many numbers: ");
    scanf("%d", &n);
    ptr = (float *)malloc(n * sizeof(float));
    if (ptr == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }
    printf("Enter %d numbers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%f", &ptr[i]);
        sum += ptr[i];
    }
```

```
    average = sum / n;
    printf("Sum = %.2f\nAverage = %.2f\n",
sum, average);
    free(ptr);
    return 0;
}

// Q4: Stack using Dynamic Memory
Allocation
#include <stdio.h>
#include <stdlib.h>

int main_stack() {
    int *stack, n, top = -1, choice, value, i;
    printf("Enter size of stack: ");
    scanf("%d", &n);
    stack = (int *)malloc(n * sizeof(int));
    if (stack == NULL) {
        printf("Memory allocation failed\n");
        return 1;
```

```
    }
    do {
        printf("\n--- Stack Menu ---\n");
        printf("1. Push\n2. Pop\n3.
Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                if (top == n - 1)
                    printf("Stack Overflow\n");
                else {
                    printf("Enter value to push: ");
                    scanf("%d", &value);
                    top++;
                    stack[top] = value;
                }
                break;
            case 2:
                if (top == -1)
```

```
            printf("Stack Underflow\n");
        else {
            printf("Popped element:
%d\n", stack[top]);
            top--;
        }
        break;
    case 3:
        if (top == -1)
            printf("Stack is Empty\n");
        else {
            printf("Stack elements:\n");
            for (i = top; i >= 0; i--)
                printf("%d\n", stack[i]);
        }
        break;
    case 4:
        printf("Exiting program...\n");
        break;
    default:
```

```
            printf("Invalid choice\n");
        }
    } while (choice != 4);
    free(stack);
    return 0;
}

// Q5: Read and Display File Contents
#include <stdio.h>
#include <stdlib.h>

int main_readfile() {
    FILE *fp;
    char ch;
    fp = fopen("filename.txt", "r");
    if (fp == NULL) {
        printf("File cannot be opened\n");
        return 1;
    }
    printf("File contents:\n");
```

```
    while ((ch = fgetc(fp)) != EOF) {
        putchar(ch);
    }
    fclose(fp);
    return 0;
}

// Q6: Append Text to a File
#include <stdio.h>
#include <stdlib.h>

int main_appendfile() {
    FILE *fp;
    char text[200];
    fp = fopen("filename.txt", "a");
    if (fp == NULL) {
        printf("Unable to open file\n");
        return 1;
    }
    printf("Enter text to append: ");
```

```
    getchar();
    fgets(text, sizeof(text), stdin);
    fputs(text, fp);
    fclose(fp);
    printf("Text appended successfully\n");
    return 0;
}

// Q7: Read Student Records and Display Rank-wise
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Student {
    char name[50];
    int roll;
    float marks;
};
```

```
int main_students() {
    struct Student s[100], temp;
    FILE *fp;
    int count = 0, i, j;
    fp = fopen("students.dat", "r");
    if (fp == NULL) {
        printf("File cannot be opened\n");
        return 1;
    }
    while (fscanf(fp, "%s %d %f",
s[count].name, &s[count].roll,
&s[count].marks) == 3) {
        count++;
    }
    fclose(fp);
    for (i = 0; i < count - 1; i++) {
        for (j = i + 1; j < count; j++) {
            if (s[i].marks < s[j].marks) {
                temp = s[i];
                s[i] = s[j];
```

```
                s[j] = temp;
            }
        }
    }
    printf("\nRank-wise Student List:\n");
    for (i = 0; i < count; i++) {
        printf("Rank %d: Name: %s | Roll: %d
| Marks: %.2f\n", i + 1, s[i].name, s[i].roll,
s[i].marks);
    }
    return 0;
}
```