Section A

1. (SAaSum)

```
#include <stdio.h>
#include <math.h>
float aSum(int n, float vector[]);
int main()
{
   float vector[10];
   int i, n;

   printf("Enter vector size: ");
   scanf("%d", &n);
   printf("Enter %d data: ", n);
   for (i=0; i<n; i++)
      scanf("%f", &vector[i]);
   printf("aSum(): %f", aSum(n, vector));
   return 0;
}
/* write your code here */
float aSum(int size, float vector[])
{
   float sum = 0;
   int j;

   for (j = 0; j < size; j++){
      sum += fabs(vector[j]);
   }
   return sum;
}
```

2. (SAreverseAr)

```
#include <stdio.h>
void reverseAr(int ar[ ], int size);
int main()
{
   int ar[20], i, size;
   printf("Enter array size: ");
   scanf("%d", &size);
   printf("Enter %d data: ", size);
   for (i=0; i <= size-1; i++)
      scanf("%d", &ar[i]);
   reverseAr(ar, size);
   printf("reverseAr(): ");
   if (size > 0) {
      for (i=0; i<size; i++)
         printf("%d ", ar[i]);
   }
   return 0;
}
/* write your code here */
void reverseAr(int ar[ ], int size)
{
   int i, temp;
   if (size > 0){
      for (i=0; i<size/2; i++){
         temp = ar[i];
         ar[i] = ar[size-i-1];
         ar[size-i-1] = temp;
      }
   }
}
```

3. (SAfindAr)

```
#include <stdio.h>
int findAr(int size, int ar[], int target);
int main()
{
   int ar[20];
   int size, i, target;

   printf("Enter array size: ");
   scanf("%d", &size);
   printf("Enter %d data: ", size);
   for (i=0; i<=size-1; i++)
      scanf("%d", &ar[i]);
   printf("Enter the target number: ");
   scanf("%d", &target);
   printf("findAr(): %d",
      findAr(size, ar, target));
   return 0;
}
/* write your code here */
int findAr(int size, int ar[], int target) {
   int j;
   for (j = 0; j < size; j++){
      if (ar[j] == target)
         return j;
   }
   return -1;
}
```

4. (SAswap2Rows)

```
#include <stdio.h>
#define SIZE 3
void swap2Rows(int ar[SIZE][SIZE], int r1, int r2);
void display(int ar[SIZE][SIZE]);
int main()
{
   int ar[SIZE][SIZE];
   int row1, row2;
   int i,j;

   printf("Enter the matrix row by row: \n");
   for (i=0; i<SIZE; i++)
      for (j=0; j<SIZE; j++)
         scanf("%d", &ar[i][j]);
   printf("Enter two rows for swapping: ");
   scanf("%d %d", &row1, &row2);
   swap2Rows(ar, row1, row2);
   printf("The array is: \n");
   display(ar);

   return 0;
}
void display(int M[SIZE][SIZE])
{
   int l,m;
   for (l = 0; l < 3; l++) {
      for (m = 0; m < 3; m++)
         printf("%d ", M[l][m]);
      printf("\n");
   }
}
/* write your code here */
void swap2Rows(int M[SIZE][SIZE], int r1, int r2)
{
   int temp;
   int n;
   for(n = 0; n < SIZE; n++){
      temp = M[r1][n] ;
      M[r1][n] = M[r2][n];
      M[r2][n] = temp;
   }
}
```

5. (SAswap2Cols)

```
#include <stdio.h>
#define SIZE 3
void swap2Cols(int ar[SIZE][SIZE], int c1, int c2);
void display(int ar[SIZE][SIZE]);

int main()
{
   int ar[SIZE][SIZE];
   int col1, col2;
   int i,j;

   printf("Enter the matrix row by row: \n");
   for (i=0; i<SIZE; i++)
      for (j=0; j<SIZE; j++)
         scanf("%d", &ar[i][j]);
   printf("Enter two columns for swapping: ");
   scanf("%d %d", &col1, &col2);
   swap2Cols(ar, col1, col2);
   printf("The array is: \n");
   display(ar);
   return 0;
}
void display(int M[SIZE][SIZE])
{
   int l,m;
   for (l = 0; l < 3; l++) {
      for (m = 0; m < 3; m++)
         printf("%d ", M[l][m]);
      printf("\n");
   }
}
/* write your code here */
void swap2Cols(int M[SIZE][SIZE], int c1, int c2)
{
   int temp;
   int n;
   for(n = 0; n < SIZE; n++){
      temp = M[n][c1] ;
      M[n][c1] = M[n][c2];
      M[n][c2] = temp;
   }
}
```

6. (SAminMax)

```
#include <stdio.h>
void minMax(int a[5][5], int *min, int *max);
int main()
{
   int A[5][5];
   int i, j;
   int min, max;

   printf("Enter the matrix (5x5) row by row: \n");
   for (i=0; i<5; i++)
      for (j=0; j<5; j++)
         scanf("%d", &A[i][j]);
   minMax(A, &min, &max);
   printf("min = %d; max = %d", min, max);
   return 0;
}
/* write your code here */
void minMax(int a[5][5], int *min, int *max)
{
   int i, j;
   *max = a[0][0];
   *min = a[0][0];
   for (i=0; i<5; i++){
      for (j=0; j<5; j++){
         if (a[i][j] > *max)
            *max = a[i][j];
         else if (a[i][j] < *min)
            *min = a[i][j];
      }
   }
}
```

Section B

1. (SBprocessString)

```
#include <stdio.h>
#include <string.h>
void processString(char *str, int size);
int main()
{
   char str[50];
   int size;

   printf("Enter a string: ");
   gets(str);
   size = strlen(str);
   processString(str, size);
   return 0;
}
/* write your code here */
void processString(char *str, int size)
{
   int totVowels = 0, totDigits = 0;
   int i;

   for (i=0; i < size; i++){
      if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i' ||
str[i] == 'o' || str[i] == 'u' || str[i] == 'A' || str[i] ==
'E' || str[i] == 'I' || str[i] == 'O' || str[i] == 'U')
         totVowels++;
       else if (str[i] >= '0' && str[i] <= '9')
         totDigits++;
   }
   printf("Total vowels = %d\n", totVowels);
   printf("Total digits = %d\n", totDigits);
}
```

2. (SBstringncpy)

```
#include <stdio.h>
char *stringncpy(char *s1, char *s2, int n);
int main()
{
   char sourceStr[40] = "source";
   char targetStr[40], *target;
   int length;

   printf("Enter a string: ");
   gets(sourceStr);
   printf("Enter no. of characters: ");
   scanf("%d", &length);
   target = stringncpy(targetStr, sourceStr, length);
   printf("stringncpy(): %s", target);
   return 0;
}
/* write your code here */
char *stringncpy(char *s1, char *s2, int n)
{
   int k, h;

   for (k = 0; k < n; k++){
      if (s2[k] != '\0')
         s1[k] = s2[k];
      else
         break;
   }
   s1[k] = '\0';

   for (h = k; h < n; h++){
      s1[h] = '\0';
   }
   return s1;
}
```

3. (SBfindTarget)

```
#include <stdio.h>
#define SIZE 10
int findTarget(char *target, char nameptr[SIZE][80], int
size);
int main()
{
   char nameptr[SIZE][80];
   char t[40];
   int i, result, size;

   printf("Enter no. of names: ");
   scanf("%d", &size);
   printf("Enter %d names: ", size);
   for (i=0; i<size; i++)
      scanf("%s", nameptr[i]);
   printf("Enter target name: ");
   scanf("\n");
   gets(t);
   result = findTarget(t, nameptr, size);
   printf("findTarget(): %d\n",  result);
   return 0;
}
/* write your code here */
```

```
int findTarget(char *target, char nameptr[SIZE][80], int
size) {
   int x, y, count, same, found = 0;
   for (x = 0; x < size; x++) {
      count = same = 0;
      while (nameptr[x][count] != '\0' || *(target +
         count) != '\0') {
         if (*(target + count) == nameptr[x][count]) {
            same += 1;
         }
         else {
            same = -1;
            break;
         }
         count += 1;
      }

      if (count == same) {
         found = x;
         break;
      }
      found = -1;
    }
    return found;
}
```

4. (SBintersect)

```
#include <stdio.h>
#include <math.h>
struct circle {
   double radius;
   double x;
   double y;
};
int intersect(struct circle c1, struct circle c2);
int main()
{
   struct circle c1, c2;
   printf("Enter circle 1 (radius x y): ");
   scanf("%lf %lf %lf", &c1.radius, &c1.x, &c1.y);
   printf("Enter circle 2 (radius x y): ");
   scanf("%lf %lf %lf", &c2.radius, &c2.x, &c2.y);
   printf("intersect(): %d\n", intersect(c1, c2));
   return 0;
}
/* write your code here */
int intersect(struct circle c1, struct circle c2)
{
   double a, b;
   int result;

   a = c1.x - c2.x;
   b = c1.y - c2.y;
   return (sqrt(a*a + b*b) <= (c1.radius + c2.radius));
}

int intersect(struct circle c1, struct circle c2) {
   float distance = sqrt(pow((c2.x - c1.x), 2) + pow((c2.y -
c1.y), 2));
   return (distance <= (c2.radius + c1.radius));
}
```

5. (SBcontain)

```
#include <stdio.h>
#include <math.h>
struct circle {
   double radius;
   double x;
   double y;
};
int contain(struct circle *c1, struct circle *c2);
int main()
{
   struct circle c1, c2;

   printf("Enter circle 1 (radius x y): ");
   scanf("%lf %lf %lf", &c1.radius, &c1.x, &c1.y);
   printf("Enter circle 2 (radius x y): ");
   scanf("%lf %lf %lf", &c2.radius, &c2.x, &c2.y);
   printf("contain(): %d\n", contain(&c1, &c2));
   return 0;
}
/* write your code here */
int contain(struct circle *c1, struct circle *c2)
{
   double a, b;

   a = c1->x - c2->x;
   b = c1->y - c2->y;
   return (c1->radius >= (c2->radius + sqrt(a * a + b * b)));
}

int contain(struct circle *c1, struct circle *c2) {
   float distance = sqrt(pow((c2->x - c1->x), 2) + pow((c2->y
- c1->y), 2));
   return(c1->radius >= (c2->radius + distance));
}
```

6. (SBcompute2)

```
#include <stdio.h>
typedef struct {
   float operand1, operand2;
   char op;
} bexpression;
float compute2(bexpression *expr);
int main()
{
   bexpression e;
   printf("Enter expression (op1 op2 op) : ");
   scanf("%f %f %c", &e.operand1, &e.operand2, &e.op);
   printf("compute2(): %f\n", compute2(&e));
   return 0;
}
/* write your code here */
float compute2(bexpression *expr)
{
   switch (expr->op) {
      case '+': return expr->operand1 + expr->operand2;
         break;
      case '-': return expr->operand1 - expr->operand2;
          break;
      case '*': return expr->operand1 * expr->operand2;
         break;
      case '/': return expr->operand1 / expr->operand2;
         break;
      default: return 0;
   }
}
```

Section C

1. (SCrSumUp2)

```
#include <stdio.h>
void rSumUp2(int n, int *result);
int main()
{
   int n, result;

   printf("Enter a number: ");
   scanf("%d", &n);
   rSumUp2(n, &result);
   printf("rSumUp2(): %d", result);
   return 0;
}
/* write your code here */
void rSumUp2(int n, int *result)
{
   if (n == 1){
      *result = 1;
   }
   else{
      rSumUp2(n-1, result);
      *result += n;
   }
}
```

2. (SCrNumDigits1)

```
#include <stdio.h>
int rNumDigits1(int num);
int main()
{
   int number;

   printf("Enter a number: ");
   scanf("%d", &number);
   printf("rNumDigits1(): %d\n", rNumDigits1(number));
   return 0;
}
/* write your code here */
int rNumDigits1(int n)
{
   if (n < 10){
      return 1;
   }
   else{
      return rNumDigits1(n/10) + 1;
   }
}
```

3. (SCrDigitPos2)

```
#include <stdio.h>
void rDigitPos2(int num, int digit, int *pos);
int main()
{
   int number;
   int digit, result=0;

   printf("Enter a number: ");
   scanf("%d", &number);
   printf("Enter the digit: ");
   scanf("%d", &digit);
   rDigitPos2(number, digit, &result);
   printf("rDigitPos2(): %d", result);
   return 0;
}
/* write your code here */
void rDigitPos2(int num, int digit, int *pos)
{
   if (num % 10 == digit){
      *pos = 1;
   }
   else if (num < 10){
      *pos = 0;
   }
   else{
      rDigitPos2(num/10, digit, pos);
      if (*pos > 0){
         *pos = *pos + 1;
      }
      else{
         *pos = 0;
      }
   }
}
```

4. (SCrDigitValue1)

```
#include  <stdio.h>
int rDigitValue1(int, int);
int main()
{
   int k;
   int number, pos, digit;

   printf("Enter a number: ");
   scanf("%d", &number);
   printf("Enter the position: ");
   scanf("%d", &k);
   printf("rDigitValue1(): %d\n", rDigitValue1(number, k));
   return 0;
}
/* write your code here */
int rDigitValue1(int n, int k)
{
   if (k==0){
      return 0;
   }
   if (k==1){
      return n%10;
   }
   return rDigitValue1(n/10, k-1);
}
```

5. (SCrSquare2)

```
#include  <stdio.h>
void rSquare2(int num, int *result);
int main()
{
   int x, result;

   printf("Enter a number: ");
   scanf("%d", &x);
   rSquare2(x, &result);
   printf("rSquare2(): %d", result);
   return 0;
}
/* write your code here */
void rSquare2(int num, int *result)
{
   if (num == 1){
      *result = 1;
   }
   else{
      rSquare2(num-1, result);
      *result += (2*num-1);
   }
}
```

6. (SCrCountArray)

```
#include <stdio.h>
#define SIZE 10
int rCountArray(int ar[], int n, int a);
int main()
{
   int ar[SIZE];
   int index, count, target, size;

   printf("Enter array size: ");
   scanf("%d", &size);
   printf("Enter %d numbers: ", size);
   for (index = 0; index < size; index++)
      scanf("%d", &ar[index]);
   printf("Enter the target: ");
   scanf("%d", &target);
   count = rCountArray(ar, size, target);
   printf("rCountArray(): %d", count);
   return 0;
}
/* write your code here */
int rCountArray(int ar[], int n, int a)
{
   if (n == 1){
      if (ar[0] == a)
         return 1;
      else
         return 0;
   }
   if (ar[0] == a)
      return 1 + rCountArray(&ar[1], n-1, a);
   else
      return rCountArray(&ar[1], n-1, a);
}
```

Section D

1. (SDcomputeMatrix)

```
#include <stdio.h>
void computeM(float matrix[4][4]);
int main(){
   float ar[4][4];
   int i,j;

   printf("Input data: \n");
   for (i = 0; i < 4; i++) {
      for (j = 0; j < 4; j++)
         scanf("%f", &ar[i][j]);
   }
   computeM(ar);
   printf("Output:\n");
   for (i = 0; i < 4; i++) {
      for (j = 0; j < 4; j++)
         printf("%.2f ", ar[i][j]);
      printf("\n");
   }
   return 0;
}
/* write your code here */
void computeM(float matrix[4][4]) {
   int x, y, sum;
   for (x = 0; x < 4; x++) {
      sum = 0;
      for (y = 0; y < 4; y++) {
         if (y == 3) {
            matrix[x][y] = sum / 3;
         }
         else {
            sum += matrix[x][y];
         }
      }
   }
}
```

2. (SDpalindrome)

```
#include <stdio.h>
int palindrome(char *str);
int main(){
   char str[80];

   printf("Enter your string: ");
   gets(str);
   if (palindrome(str))
      printf("palindrome(): A palindrome.\n");
   else
      printf("palindrome(): Not a palindrome.\n");
   return 0;
}
/* write your code here */
int palindrome(char *str) {
   int x, count = 0, same = 1;
   while (*(str + count) != '\0') {
      count++;
   }
   for (x = 0; x < count / 2; x++) {
      if (*(str + x) != *(str + count - x - 1)) {
         same = 0;
         break;
      }
   }
   return same;
}
```

3. (SDfindSubstring)

```
#include <stdio.h>
int findSubstring(char *s, char *t);
int main()
{
   char sourceStr[40], targetStr[40];

   printf("Enter a source string: ");
   gets(sourceStr);
   printf("Enter the target string: ");
   gets(targetStr);
   printf("findSubstring(): %d\n", findSubstring(sourceStr,
targetStr));
   return 0;
}
/* write your code here */
int findSubstring(char *s, char *t) {
    int x = 0, y, pos, stringFound = -1;
    while (*(t + x) != '\0') {
        if (*(t + x) == *s) {
            y = 0;
            pos = x;
            stringFound = 1;
            while (*(s + y) != '\0') {
                if (*(s + y) != *(t + pos + y)) {
                    stringFound = -1;
                    break;
                }
                y++;
            }
        }
        if (stringFound == 1) {
            break;
        }
        x++;
    }

    return stringFound;
}
```

4. (SDcustomer)

```
#include <stdio.h>
#include <string.h>
struct account {
   struct
   {
      char lastName[10];
      char firstName[10];
   } names;
   int accountNum;
   double balance;
};
void nextCustomer(struct account *acct);
void printCustomer(struct account acct);
int main()
{
   struct account record;
   int flag = 0;
   do {
      nextCustomer(&record);
      if ((strcmp(record.names.firstName, "End") == 0) &&
          (strcmp(record.names.lastName, "Customer") == 0))
         flag = 1;
      if (flag != 1)
         printCustomer(record);
   } while (flag != 1);
}
/* write your code here */
```

```
void nextCustomer(struct account *acct)
{
   printf("Enter names (firstName lastName): ");
   scanf("%s" "%s", acct->names.firstName,
   acct->names.lastName);
   if ((strcmp(acct->names.firstName, "End") != 0) &&
    (strcmp(acct->names.lastName, "Customer") != 0)) {
      printf("Enter account number: ");
      scanf("%d", &acct->accountNum);
      printf("Enter balance: ");
      scanf("%lf", &acct->balance);
   }
}
void printCustomer(struct account acct)
{
   printf("Customer record: %s %s %d %lf\n",
      acct.names.firstName, acct.names.lastName,
      acct.accountNum, acct.balance);
}
```

5. (SDrSumOddDigits1)

```
#include <stdio.h>
int rSumOddDigits1(int n);
int main()
{
   int num, result;

   printf("Enter a number: ");
   scanf("%d", &num);
   printf("rSumOddDigits1(): %d\n", rSumOddDigits1(num));
   return 0;
}
/* write your code here */
int rSumOddDigits1(int n)
{
   if(n == 0)
      return 0;
   if(n%2 == 1)
      return n%10 + rSumOddDigits1(n/10);
   else
      return rSumOddDigits1(n/10);
}
```

6. (SDrReverseAr)

```
#include <stdio.h>
void rReverseAr(int ar[], int size);
int main()
{
   int ar[10], size, i;

   printf("Enter array size: ");
   scanf("%d", &size);
   printf("Enter %d numbers: ", size);
   for (i=0; i<size; i++)
      scanf("%d", &ar[i]);
   rReverseAr(ar, size);
   printf("rReverseAr(): ");
   for (i=0; i<size; i++)
      printf("%d ", ar[i]);
   return 0;
}
/* write your code here */
void rReverseAr(int ar[], int n)
{
   if (n > 1){
      int temp = ar[n-1];
      ar[n-1] = ar[n-n];
      rReverseAr(&ar[1], n-2);
      ar[n-n] = temp;
   }
}

void rReverseAr(int ar[], int size) {
   if (size > 1) {
      int temp = ar[0];
      ar[0] = ar[size - 1];
      ar[size - 1] = temp;
      rReverseAr(ar + 1, size - 2);
   }
}
```